51CTO.com
技术成就梦想

我 们 只 谈 开 发

开发月刊

Development Monthly

2012年05月
总第014期



编程行业换代快，老前辈们今何在?

“菜鸟”程序员和“大神”程序员差距在哪里

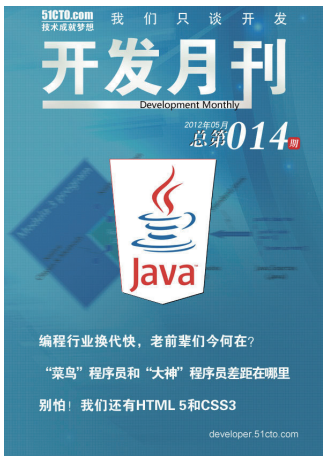
别怕！我们还有HTML 5和CSS3

developer.51cto.com

目录

2012.05

Contents



编程排行 Billboard

3 5月编程排行榜:爱立信走出Erlang

专题报道 《Java开发与JVM》

6 Java学习之路：不走弯路，就是捷径
8 Java Excel API及详细教程
9 Java Swing杂记
11 JVM的内存溢出异常

技术热点 Techlogy hot

13 宅男程序员给老婆课程10：做就对了
14 宅男程序员给老婆课程11：领域模型
16 Qcon 2012开发者关注云计算与敏捷
17 编程行业换代快，老前辈们今何在?
19 谷歌I/O 2012开发人员该关注什么
21 想要快速持续交付 何以如此困难?
23 全面掌握SQL Server Profiler系列:1
25 别怕！我们还有HTML 5和CSS3
27 面向对象的JavaScript基本知识指南
28 开发HTML5跨平台游戏相关经验
30 JS大项目代码的多人合作处理方法
31 阿里巴巴离职DBA 35岁总结的职业生涯
34 淘宝下单高并发解决方案
36 张利华：夺命的中国IT互联网行业
38 PKzip创始人Phil Katz短暂而饱受折磨的一生
41 开发者拒绝写技术博客的常见理由

51CTO.com
技术成就梦想

出版方：北京无忧创想信息技术有限公司

责任编辑：彭凡 林师授 张伟　　封面设计：钱靓

联系方法：pengfan@51cto.com　010-68476606（分机 8058）

出版日期：2012 年 5 月 16 日

欢迎广大读者投稿

5月编程排行榜:爱立信走出Erlang

2012年5月9日,Tiobe公布了最新一期编程语言排行榜。C语言继续压制Java位居第一,两者之间的差距有越拉越大的趋势。而Object-C上升势头良好,已经开始威胁C++的老三位置。本期排行榜给我们一个问题,谁会是下一个最受欢迎的编程语言? Tiobe给出5个答案,分别是Scala、F#、Groovy、Erlang和Clojure。今天我们要关注的,是从爱立信计算机科学实验室走出来的Erlang。

2012年5月9日,Tiobe公布了最新一期编程语言排行榜。C语言继续压制Java位居第一,两者之间的差距有越拉越大的趋势。而Object-C上升势头良好,已经开始威胁C++的老三位置。本期排行榜给我们一个问题,谁会是下一个最受欢迎的编程语言? Tiobe给出5个答案,分别是Scala、F#、Groovy、Erlang和Clojure。今天我们要关注的,是从爱立信计算机科学实验室走出来的Erlang。往期编程语言排行榜点这里。

大家先看2012年5月编程语言排行榜榜单

Position May 2012	Position May 2011	Delta in Position	Programming Language	Ratings May 2012	Delta May 2011	Status
1	2	↑	C	17.346%	+1.18%	A
2	1	↓	Java	16.599%	-1.56%	A
3	3	=	C++	9.825%	+0.68%	A
4	6	↑↑	Objective-C	8.309%	+3.30%	A
5	4	↓	C#	6.823%	-0.72%	A
6	5	↓	PHP	5.711%	-0.80%	A
7	8	↑	(Visual) Basic	5.457%	+0.96%	A
8	7	↓	Python	3.819%	-0.76%	A
9	9	=	Perl	2.805%	+0.57%	A
10	11	↑	JavaScript	2.135%	+0.74%	A
11	10	↓	Ruby	1.451%	+0.03%	A
12	26	↑↑↑↑↑↑↑↑↑↑	Visual Basic .NET	1.274%	+0.79%	A
13	21	↑↑↑↑↑↑↑↑	PL/SQL	1.119%	+0.62%	A
14	13	↓	Delphi/Object Pascal	1.004%	-0.07%	A
15	15	=	Lisp	0.941%	-0.01%	A
16	24	↑↑↑↑↑↑↑↑	Logo	0.839%	+0.35%	A-
17	17	=	Pascal	0.808%	+0.10%	A
18	18	=	Transact-SQL	0.654%	-0.04%	A-
19	16	↓↓↓	Ada	0.649%	-0.10%	B
20	12	↓↓↓↓↓↓↓↓	Lua	0.566%	-0.54%	B

对于爱立信(Ericsson),大家首先想到是那个大名鼎鼎的手机厂商。在上世纪九十年代,爱立信的地位与当初的诺基亚和今天的苹果一样,用户数量巨大。不过英雄也有落幕之时,现在的索爱早已不如当年的爱立信那般风光了。

Erlang专为电信行业而生

在Erlang出生之前,软件行业中没有一款编程语言包容电信行业所需要的所有的特性。为了下一代电信产品的研发,在20世纪80年代中期,爱立信的计算机科学实验室接到为下一代电信产品寻找合适语言的任务。Joe Armstrong、Robert Virding和Mike Williams在Bjarne Dcker的带领下,他们两年里尝试了所有的开发语言。在测试的过程中,实验室发现尽管有一些语言在某些方面符合电信行业的要求,但都无法覆盖所有的特性。看来很有必要创造一门编程语言来满足需求。

从此Erlang诞生了,它受到了函数语言(比如ML和Miranda),并发语言(比如ADA、Modula、Chill)以及逻辑编程语言Prolog语言的启发和影响。与爱立信专有语言EriPascal和PLEX一样,Smalltalk语言的软件升级特性在Erlang中也得到了深刻的体现。

1991年,Mike Williams终于写成了以C语言为基础的Erlang虚拟机的最初版本,一年之后,第一个应用Erlang的商业项目也在一个小团队的带领下诞生了。该项目是一个移动服务器,它允许DECT无线电话的用户漫游于私人办公室网络。该产品在1994年成功问世,并随后得到了用户很多关于技术改进和新特性的宝贵反馈,

2012 年 5 月编程语言排行榜：从爱立信走出来的 Erlang II

这些建议都在 1995 年的 Erlang 新的版本中得以实现。

Erlang 到底能做什么?

虽然 Erlang 本身是一种很有吸引力的编程语言，但当你把它与虚拟机（VM）、OTP 中间件和类库相结合的时候，其真正的实力才能完全体现出来。其中的每一点都使 Erlang 软件开发变得如此特别。那么，具体有哪些功能体现出 Erlang 与其他类似语言的不同呢?

高级构造

Erlang 是一种声明性的语言。声明性语言工作的原则是去描述应该计算什么，而不是去解释这个值是如何计算而来的。一个函数定义就像一组等式，尤其是当使用模式匹配从不同的情况中去选择和从复杂的数据结构中抽取组件的时候。下面是一个简单的例子：

```
area({square, Side}) -> Side * Side ;
area({circle, Radius}) -> math:pi() * Radius * Radius.
```

可扩展、安全和高效的并发

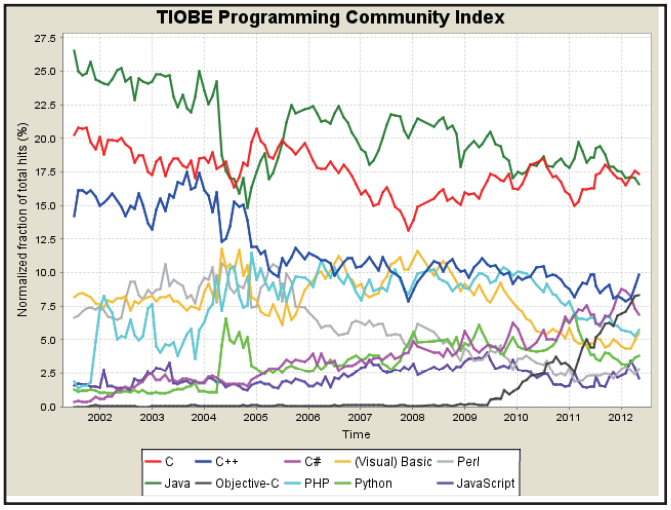
Erlang 的并发具有快速和可扩展的特性。它的进程是轻量级的，Erlang 虚拟机不会为每一个已生成的进程创建一个操作系统线程。Erlang 进程在虚拟机中生成、调度和处理，而与底层的操作系统无关。因此，进程的生成时间是以微秒为单位的，并且独立于现存的进程的数量。比较而言，Java 和 C# 为每一个进程生成一个底层的操作系统线程：由此你能得出一个有竞争性的结论是，Erlang 在这方面以绝对的优势胜过其他两种语言。

全文：

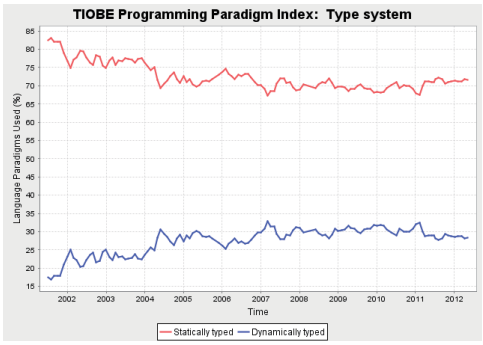
http://developer.51cto.com/art/201205/334986.htm

下面是本期编程语言排行榜的其他排名数据和趋势走向。

前 10 名编程语言走势图



Bash, bc, BETA, BlitzMax, Boo, Bourne shell, C++/CLI, CL (OS/400), Clean, Clojure, cT, Dart, Dylan, Eiffel, Euphoria, Factor, Icon, Inform, Io, J, JScript.NET, Korn shell, LabVIEW, LabWindows/CVI, Ladder Logic, Lingo, Max/MSP, Modula-2, MUMPS, NATURAL, Oberon, Occam, OpenCL, Oz, PILOT, PowerShell, Revolution, REXX, S, S-PLUS, SAS, Scala, VBScript, VHDL, Z shell





Java 是一门不算年轻的语言。针对 Java 开发的资料汗牛充栋，51CTO 这一期《开发月刊》的主题定在 Java 开发和 JVM。也是为了让 Java 程序员有一些新的内容。

在新一期的 Tiobe 编程语言排行榜上，Java 依旧被 C 语言压制。是否意味着很多公司的项目正在转变？

Java 依旧占据很大的份额，特别是 Android 日渐强大之后。但对于传统软件行业开发者来说，Java 仅仅是赖以为生的谋生手段吗？

向您推荐《JVM 编程语言，你的选择》

51CTO 专 题 链 接：http://developer.51cto.com/exp/jvm/index.html

51CTO 开发频道语

Java学习之路：不走弯路，就是捷径



JAVA 是一种平台，也是一种程序设计语言，如何学好程序设计不仅仅适用于 JAVA，对 C++ 等其他程序设计语言也一样管用。

JAVA 是一种平台，也是一种程序设计语言，如何学好程序设计不仅仅适用于 JAVA，对 C++ 等其他程序设计语言也一样管用。有编程高手认为，JAVA 也好 C 也好没什么分别，拿来就用。为什么他们能达到如此境界？我想是因为编程语言之间有共通之处，领会了编程的精髓，自然能够做到一通百通。如何学习程序设计理所当然也有许多共通的地方。

1. 如何学习程序设计？

JAVA 是一种平台，也是一种程序设计语言，如何学好程序设计不仅仅适用于 JAVA，对 C++ 等其他程序设计语言也一样管用。有编程高手认为，JAVA 也好 C 也好没什么分别，拿来就用。为什么他们能达到如此境界？我想是因为编程语言之间有共通之处，领会了编程的精髓，自然能够做到一通百通。如何学习程序设计理所当然也有许多共通的地方。

1.1 培养兴趣

兴趣是能够让你坚持下去的动力。如果只是把写程序作为谋生的手段的话，你会活的很累，也太对不起自己了。多关心一些行业趣事，多想想盖茨。不是提倡天天做白日梦，但人要是没有了梦想，你觉得有味道吗？可能像许多深圳本地农民一样，打打麻将，喝喝功夫茶，拜拜财神爷；每个月就有几万十几万甚至更多的进帐，凭空多出个 " 食利阶层 "。你认为，这样有味道吗？有空多到一些程序员论坛转转，你会发现，他们其实很乐观幽默，时不时会冒出智慧的火花。

1.2 慎选程序设计语言

男怕入错行，女怕嫁错郎。初学者选择程序设计语言需要谨慎对待。软件开发不仅仅是掌握一门编程语言了事，它还需要其他很多方面的背景知识。软件开发也不仅仅局限于某几个领域，而是已经渗透到了各行各业几乎每一个角落。

如果你对硬件比较感兴趣，你可以学习 C 语言 / 汇编语言，进入硬件开发领域。如果你对电信的行业知识及网络比较熟悉，你可以在 C/C++ 等之上多花时间，以期进入电信软件开发领域。如果你对操作系统比较熟悉，你可以学习 C/Linux 等等，为 Linux 内核开发 / 驱动程序开发 / 嵌入式开发打基础。如果你想介入到应用范围最广泛的应用软件开发（包括电子商务电子政务系统）的话，你可以选择 J2EE 或 .NET，甚至 LAMP 组合。每个领域要求的背景知识不一样。做应用软件需要对数据库等很熟悉。总之，你需要根据自己的特点来选择合适你的编程语言。

1.3 要脚踏实地，快餐式的学习不可取

先分享一个故事。

有一个小朋友，他很喜欢研究生物学，很想知道那些蝴蝶如何从蛹壳里出来，变成蝴蝶便会

Java 学习之路：不走弯路，就是捷径 II

飞。有一次，他走到草原上面看见一个蛹，便取了回家，然后看着，过了几天以后，这个蛹出了一条裂痕，看见里面的蝴蝶开始挣扎，想抓破蛹壳飞出来。这个过程达数小时之久，蝴蝶在蛹里面很辛苦地拼命挣扎，怎么也没法子走出来。这个小孩看着看着不忍心，就想不如让我帮帮它吧，便随手拿起剪刀在蛹上剪开，使蝴蝶破蛹而出。但蝴蝶出来以后，因为翅膀不够力，变得很臃肿，飞不起来。

这个故事给我们的启示是：欲速则不达。

浮躁是现代人最普遍的心态，能怪谁？也许是贫穷落后了这么多年的缘故，就像当年的大跃进一样，都想大步跨入共产主义社会。现在的软件公司、客户、政府、学校、培训机构等等到处弥漫着浮躁之气。就拿笔者比较熟悉的深圳 IT 培训行业来说吧，居然有的打广告宣称 " 参加培训，100% 就业 "，居然报名的学生不少，简直是藐视天下程序员。社会环境如是，我们不能改变，只能改变自己，闹市中的安宁，弥足珍贵。许多初学者 C++/JAVA 没开始学，立马使用 VC/JBuilder，会使用 VC/JBuilder 开发一个 HelloWorld 程序，就忙不迭的向世界宣告，" 我会软件开发了 "，简历上也大言不惭地写上 " 精通 VC/JAVA"。结果到软件公司面试时要么被三两下打发走了，要么被驳的体无完肤，无地自容。到处碰壁之后才知道捧起《C++ 编程思想》《JAVA 编程思想》仔细钻研，早知如此何必当初呀。

" 你现在讲究简单方便，你以后的路就长了 "，好象也是佛经中的劝戒。

1.4 多实践，快实践

彭端淑的《为学一首示子侄》中有穷和尚与富和尚的故事。

从前，四川边境有两个和尚，一个贫穷，一个有钱。一天，穷和尚对富和尚说：" 我打算去南海朝圣，你看怎么样？ " 富和尚说：" 这里离南海有几千里远，你靠什么去呢？ " 穷和尚说：" 我只要一个水钵，一个饭碗就够了。" 富和尚为难地说：" 几年前我就打算买条船去南海，可至今没去成，你还是别去吧！ " 一年以后，富和尚还在为租赁船只筹钱，穷和尚却已经从南海朝圣回来了。

这个故事可解读为：任何事情，一旦考虑好了，就要马上上路，不要等到准备周全之后，再去干事情。假如事情准备考虑周全了再上路的话，别人恐怕捷足先登了。软件开发是一门工程学科，注重的就是实践，" 君子动口不动手 " 对软件开发人员来讲根本就是错误的，他们提倡 " 动手至上 "，但别害怕，他们大多温文尔雅，没有暴力倾向，虽然有时候蓬头垢面的一副 " 比尔盖茨 " 样。有前辈高人认为，学习编程的秘诀是：编程、编程、再编程，笔者深表赞同。不仅要多实践，而且要快实践。我们在看书的时候，不要等到你完全理解了才动手敲代码，而是应该在看书的同时敲代码，程序运行的各种情况可以让你更快更牢固的掌握知识点。

1.5 多参考程序代码

程序代码是软件开发最重要的成果之一，其中渗透了程序员的思想与灵魂。许多人被《仙剑奇侠传》中凄美的爱情故事感动……■

本文未完，请看原文：

http://developer.51cto.com/art/201204/330124.htm

编者按

有时在 Java 开发中会操作 excel 表格，其实操作起来也特别简单。这里把前期操作步骤说一下，本文会简单的介绍一个开放源码项目：Java Excel Api，使用它大家就可以方便的操作 Excel 文件了。

Java Excel API及详细教程

有时在Java开发中会操作Excel表格，其实操作起来也特别简单。这里把前期操作步骤说一下，本文会简单的介绍一个开放源码项目：Java Excel Api，使用它大家就可以方便的操作 Excel 文件了。

首先下载好：Java Excel Api，这个文件我已经和 JAVA+Excel+API 详细教程 .pdf 一并压缩上传了，感兴趣的朋友可以下载！

下载地址：http://download.csdn.net/detail/pzhtpf/4256241

我这里用的开发平台是 Eclipse，这里我把操作简单说一下：

1， 建，立 java 项目，在这个项目在建立一个新的文件夹 lib；

2， 将 jxl.jar，即 Java Excel Ap，复制到 lib

3，然后右键点击这个 java 项目，选择 Properties

4，在左侧列表里选中 Java Build Path ，右侧选中 Libraries

5，点击 Add JARs

6， 然后去选择这个项目中 lib 文件夹中的 jxl.jar，点击确定

成功后，项目中会多一个文件夹为：Referenced Libraries

准备工作完成后，就可以去操作 excel 了，下面贴一段读取的代码：

```
import java.io.File;
import java.io.IOException;
import jxl.Cell;  import jxl.Sheet;
import jxl.Workbook;
import jxl.read.biff.BiffException;
public class mainClass {
public static void main(String args[]){
File f=new File("E:"+File.separator+"dataAnalysis"+
File.separator+"test.xls");
// 这是我自己建的，位于 E:\dataAnalysis 文件夹中的 test.xls
try {
Workbook book=Workbook.getWorkbook(f);//

Sheet sheet=book.getSheet(0);
// 获得第一个工作表对象
for(int i=0;i<sheet.getRows();i++){
for(int j=0;j<sheet.getColumns();j++){
Cell cell=sheet.getCell(j, i);
// 获得单元格
System.out.print(cell.getContents()+" ");
}
System.out.print("\n");
}    }
catch (BiffException e) {
 // TODO Auto-generated catch block
e.printStackTrace();
} catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
} } } ■
```

本文未完，请看原文：

http://developer.51cto.com/art/201204/332464.htm

Java Swing 杂记：引入安卓的 NinePatch 技术

编者按

本文诣在展示如何在 Swing 中引入 Android 的 NinePatch 技术（Android 早期有文章里中文译作九格图，暂且这么叫吧 ^_^，但此术非传统……

【摘 要】

本文诣在展示如何在 Swing 中引入 Android 的 NinePatch 技术（Android 早期有文章里中文译作九格图，暂且这么叫吧 ^_^，但此术非传统移动手机上的功能布局——九格图哦）。

【准备篇】

Q：何为 Android 的 NinePatch 技术？

A：说简单点，就是 Android 平台中用于对图片据屏幕大小进行自动拉伸的技术。更准确的介绍详见此文：http://www.yixieshi.com/ucd/9142.html。

NinePatch 技术本身虽微不足道，但它对于 Android 的 UI 定制开发来说无疑是相当有价值的。也可直接看官方指南：http://developer.android.com/guide/developing/tools/draw9patch.html。

Q：NinePatch 对 Swing 意味着什么？

A：利用 NinePatch 技术，比如：你在美化 Swing 的按钮 UI 时，再也不用根据不同的按钮大小准备不同的图片了，一张图片解决不同按钮按各自大小自动拉伸填充的问题，多么神奇！

当然，如果你对 Android 的程序比较熟，或者说对 android 的程序外观定制比较熟的话，你将会更清楚这一点——Swing 的外观定制能力将会因此变的无比灵活和强大，很多不可能将成为现实。Android 程序的外观定制其实有点 Java 标准平台换肤技术 Synth 的影子，但显然，这个聪名的小改进，使得 Android 的外观定制比 Swing 更容易、更灵活。

Q：Android 平台的 NinePatch 技术可以直接拿到 Java 标准平台上使用？

A：没错，两个 java 平台除去 UI 部份外，本身有很大一部分是可以互相兼容的，具体情况请了解 android 平台 java 技术的前世今生。

Q：从何处获得 Android 平台的 NinePatch 技术呢？

A：NinePatch 技术的核心只有 3 到 4 个类，拿过来用就可以了，源码地址可以在此链接找到：

http://www.java2s.com/Open-Source/Android/android-core/platform-sdk/com.android.ninepatch.htm，我打好的包稍

后可以在附件里下载哦。

【准备好 .9.png 图片】

本图片将使用 NinePatch 技术作为演示代码中的一 JPanel 背景进行自动填充之用，用不同的 2 张图是为了方便进行效果展示：



Java Swing 杂记：引入安卓的 NinePatch 技术 II

【测试代码】

略，见原文。

http://developer.51cto.com/art/201205/334095.htm

【运行效果图】



【附件下载】

测试代码完整 Eclipse3.5.2 工程(含 NinePatch 的 jar 包哦)：

http://files.cnblogs.com/jb2011/jb2011blog_testNinePatch.rar

【最后再嗦几句】

潜水了很多年，得益于许多无私网友的奉献，越发觉得很有必要与人分享一些东西。接下来将陆续写出“Swing 整容”系列文章，希望对需要的人有用，但因水平确实有限，不喜者还请勿喷，多谢。

有人说，Swing 很丑，这话没错，但 Swing 真的没救了？答案当然是否定的。以下 2 张图是近期为一款产品设计开发的全新外观，接下来的文章将会与它们有关，谢谢关注。图 1，没错，这



图 1，没错，这是 Swing 的界面哦，半透明立体阴影边框这张图上看不出明显效果

图 2，这也是 Swing 的界面哦



“民工甲”的“Swing 三刀”系列文章给了我最近一次 Swing 美化工作的部分灵感，非常感谢作者的无私，文章地址是：http://joshuaxiao.iteye.com/blog/707514。

“WilliamChen”的 Swing 技术文章也是相当不错，可惜多年不更新了，有兴趣的朋友可以去围观哦，http://blog.sina.com.cn/swingjava



编者按

在 Java 虚拟机规范的描述中，除了 PC(程序计数器) 寄存器外，虚拟机内存的其他几个运行时区域都有发生 OutOfMemoryError 异常的可能。当发生 OutOfMemoryError 异常时，无法用 try...catch 捕捉。

JVM的内存溢出异常

在 Java 虚拟机规范的描述中，除了 PC(程序计数器) 寄存器外，虚拟机内存的其他几个运行时区域都有发生 OutOfMemoryError 异常的可能。当发生 OutOfMemoryError 异常时，无法用 try...catch 捕捉。

在开始讲解之前，在这里先简单介绍下虚拟机启动相关的一些内存设置参数。因为 OutOfMemoryError 异常发生，与这些参数的设置密切相关。

举例说明含义：

-Xss128k

每个线程的 java 栈大小，一个线程 java 栈所有栈帧大小总和最大允许的尺寸 128k。

-Xms128m

表示 JVM Heap(堆内存) 最小尺寸 128MB，初始分配

-Xmx512m

表示 JVM Heap(堆内存) 最大允许的尺寸 256MB，按需分配。

-XX:PermSize=20M

设置方法区的初始大小

-XX:MaxPermSize=30M

设置方法区的最大值

Java 栈溢出

在 Java 虚拟机规范中，对这个区域规定了两种异常状况：StackOverflowError 和 OutOfMemoryError 异常。

1.StackOverflowError 异常

每当 java 程序代码启动一个新线程时，Java 虚拟机都会为它分配一个 Java 栈。Java 栈以帧为单位保存线程的运行状态。当线程调用 java 方法时，虚拟机压入一个新的栈帧到该线程的 java 栈中。只要这个方法还没有返回，它就一直存在。如果线程的方法嵌套调用层次太多 (如递归调用)，随着 java 栈中帧的逐渐增多，最终会由于该线程 java 栈中所有栈帧大小总和大于 -Xss 设置的值，而产生 StackOverflowError 内存溢出异常。例子如下：

```
/**   * VM Args: -Xss128k   */
public class Test {
private int count = 0;
public static void main(String[] args) {
new Test().method();
}
public void method() {
System.out.println(++count);
method();
}   }
```

JVM 的内存溢出异常 II

-Xss 为 128k。其中的一次测试结果为，当 count 的值累加到 2312 时，发生如下异常：

```
Exception in thread "main" java.lang.StackOverflowError at sun.nio.cs.UTF_8.updatePositions(UTF_8.java:58) at sun.nio.cs.UTF_8$Encoder.encodeArrayLoop(UTF_8.java:392) at sun.nio.cs.UTF_8$Encoder.encodeLoop(UTF_8.java:447) at java.nio.charset.CharsetEncoder.encode(CharsetEncoder.java:544) at sun.nio.cs.StreamEncoder.implWrite(StreamEncoder.java:252) at sun.nio.cs.StreamEncoder.write(StreamEncoder.java:106) at java.io.OutputStreamWriter.write(OutputStreamWriter.java:190) at java.io.BufferedWriter.flushBuffer(BufferedWriter.java:111) at java.io.PrintStream.write(PrintStream.java:476) at java.io.PrintStream.print(PrintStream.java:547) at java.io.PrintStream.println(PrintStream.java:686) at jvm.Test.method(Test.java:17)
```

修改 -Xss 为 1280k。其中的一次测试结果为，当 count 的值累加到 26888 时，发生 StackOverflowError 异常。随着 -Xss 参数值的增大，可以嵌套的方法调用层次也相应增加。

综上所述，StackOverflowError 异常是由于方法调用的层次太深，最终导致为某个线程分配的所有栈帧大小总和大于 -Xss 设置的值，从而发生 StackOverflowError 异常。

2.OutOfMemoryError 异常

java 程序代码启动一个新线程时，没有足够的内存空间为该线程分配 java 栈（一个线程 java 栈的大小由 -Xss 参数确定），jvm 则抛出 OutOfMemoryError 异常。例子如下：

```
/**  * VM Args: -Xss128k  */
public class Test {
public static void main(String[] args) {
int count = 0;
while (true) {
Thread thread = new Thread(new Runnable() {
public void run() {
while (true) {
try {
Thread.sleep(5000);
} catch (Exception e) {}
}
}
});
thread.start();
System.out.println(++count);
}
}
}
```

-Xss 为 128k。其中的一次测试结果为，当 count 的值累加到 11887 时，发生如下异常：

```
Exception in thread "main" java.lang.OutOfMemoryError: unable to create new native thread at java.lang.Thread.start0(Native Method) at java.lang.Thread.start(Thread.java:640) at jvm.Test.main(Test.java:20)
```

修改 -Xss 为 1280k。其中的一次测试结果为，当 count 的值累加到 1270 时，发生 OutOfMemoryError 异常。随着 -Xss 参数值的增大，java 程序可以创建的总线程数越少。

Java 堆溢出

Java 堆用于储存对象实例。当需要为对象实例分配内存，而堆的内存占用又已经达到 -Xmx 设置的最大值……■

本文未完，请看原文：

http://developer.51cto.com/art/201205/336013.htm

宅男程序员给老婆课程10:做，就对了

学以致用，很多时候，学习一样东西最好需要能够在实际中应用起来。

所以我在第 2 课 " 怎么看待牛人 " 中强调的必须 “看代码 + 写代码”。

不过我在里面提到的例子 “ORM” 却并不好，ORM 太过庞大。实际编码，应该是从小开始。

运维工作更经常使用脚本语言，脚本程序甚至是 shell 命令都可以完成很多有意义的事情。

这些猪头应该在工作中体验很多；但作为程序员，程序能够发挥的作用也可以体现在生活上。

玩 Draw Something 单词想不出来，是完全可以写个程序来输出单词列表的。

上网下载一个英文单词词库；然后甚至可以用最傻 X 的方式去逐个单词检查，看 Draw Something 给出的字母是否能够组成各个单词。

程序首先是要完成需求，这里的需求仅仅是要方便玩游戏，猜出朋友的单词谜语。

程序运行慢点无所谓，千分之一秒输出结果，还是 10 秒输出结果，都不会影响需求实现。

（当然，如果是玩 Facebook 上的限时拚单词游戏那需求又是不同。）

这种 “程序” 是所谓 Throw-away code，写完就扔。

像 Draw Something 这样的游戏，乐趣就在于努力去想、努力猜成功之后的成就感。有了这样一个程序，那就不用努力去想，游戏的乐趣也就会在瞬间丧失，“破解工具” 自然也就得扔掉了。

即便写完就扔，但写这样程序却有其意义。写与不写差别是 0 与 1 的差别，这是本质的区别。

我会非常鄙视那些热衷于看各种语言的介绍但却一行程序都不写的人。

有的人，听说 Erlang 很牛 B，上网搜了一堆介绍，不断的感叹 “哇～ Erlang 确实很牛！”，“哦耶！Facebook Chat 跟 Web QQ 都是在用 Erlang，果然 Erlang 才是王道！”

但是，他自己却不写任何一行 erlang 程序；有时，还会抱怨公司的管理层都是傻逼，这个项目用 erlang 再合适不过，为什么不用，为什么不给团队使用 Erlang 的机会呢？

一定要写程序，没有机会，也要创造机会。

而在我看来，生活中这种 “玩游戏” 的机会再合适不过。

写了 Draw Something 的 “破解工具”，会使得猜单词没有成就感，丧失游戏的乐趣；但，完成了一个程序去破解一个游戏，这本身也是一件有成就感的事情啊～

并且，游戏的乐趣会转移为编程的乐趣；而乐趣，是让自己变厉害的最大动力。

Geek 享受这样的机会；而 ED 则等待别人享受这样的机会。

“做，就对了” – 慈济宗创始人 证严法师

作业：

1. 使用 Perl 实现一个程序输入若干字母，输出这些字母所能组成的所有单词列表。就是要写个 Draw Something 的 “破解工具”。

2. 比较 Perl 实现跟云风 lua 实现有何不同■

宅男程序员给老婆课程之11:领域模型

这个系列来自一位宅男程序员,这个系列是他写给老婆的电脑课程。以下,开始本系列的第 12 篇——域模型。



在前面的课程中提到 PEAA 中只有一页半的 Active Record Pattern (http://martinfowler.com/eaaCatalog/activeRecord.html) 影响了过去 5 年多 6 年的 Web 开发潮流。

这个潮流是由 Ruby On Rails 引领的。

RoR 的作者 DHH David Heinemeier Hansson 是 Hacker, 他因为 RoR 在 2005 年被 Google 跟 O'Reilly 选为年度黑客。

他在设计 RoR 时,选用了 Active Record 作为 RoR 的 M 层。

Active Record非常简单,一个类对应一个表,一个对象实例对应一行数据;并且有简单的有 Save / Delete 以及查询等简单的函数操作。

严格的说, Active Record 不是福勒所推崇的充血 Domain Object 模型 (http://martinfowler.com/bliki/AnemicDomainModel.html), Active Record 对象提供的功能函数太少,只有通用的数据操作方法,而不包涵业务逻辑;但它又不像 POJO (http://martinfowler.com/bliki/POJO.html) 那样完全的贫血。

(充血、贫血 Domain Object 之争,可以去 iteye 翻帖子)

从福勒 AnemicDomainModel 一文看,他在当年(2003)是推荐了充血 Domain 对象跟 POJO,但过去几年在 Web 开发领域所流行的却是 Active Record 这样两边都沾点,但却又不全是的中间妥协方案。

不搞教条主义,什么实用用什么, POJO 不够,那么就加一点;充血太复杂,那么减少一点。

从互联网的发展看,我一时间完全想不出有什么在理论上被设计得很好的模型,能够最终经历时间考验成为事实标准。

因特网的 7 层模型,实际用到的远不到 7 层;Java 的 EJB 挂了;XML 被 JSON 取代等等等等。

也许学院派提出的理论有他们的应用场景,只是,这样的场景,在快速发展互联网似乎很难找到例子。

互联网产品的业务相对简单, Active Record 已经足够好,足够方便,因此大行其道。

另一方面,互联网产品做大后,也往往有着极大的性能要求,一个复杂的模型,是难以做性能优化的。

像 Active Record,因为足够简单, Twitter 在当年遇到性能问题的时候,便直接 Hack 掉 RoR 的实现,增加了 Cache-Money (https://github.com/nkallen/cache-money) 这么一个透明的缓存层。

如果 RoR 使用的是充血对象模型,对象中有复杂的业务逻辑,如何增加透明的缓存呢?

Active Record 的实际上是对数据库操作做

了抽象。

封装、抽象是一门艺术。

什么该封装，什么该暴露，什么彻底不可见，需要拿捏得很准确。

最容易犯的错误是过度封装，使得一些本来很简单的底层操作，到了上层变得完全不能用；或者说，很难用。

开发者需要用到 hack 的方式，才能去做这些简单的操作。

Active Record 便是一个抽象封装得恰到好处的例子。过度设计、过度封装的数据操作层？EJB。

按照教科书对 OO 的定义，OO 的核心特性之一是：encapsulation http://en.wikipedia.org/wiki/Encapsulation_%28object-oriented_programming%29

Private 属性、方法，对象外部完全不能访问。

但如果遇到了需要访问的场景怎么办?!

有的人会说："这样的场景本来就不应该出现，这是对象设计一开始没有做好造成的，错误的应该设成 Public 的属性设成了 Private"。

ORM，采用 O => R 的映射的设计哲学，只考虑业务对象，完全不考虑底层数据库，数据库仅仅是一个可以被替换掉的持久层，它可以是关系型数据库、也可以是 NoSQL，甚至是硬盘文件。

也就是说，Domain Object 是把后端数据库给设成 "Private" 了，即便底层是关系型数据库，你也不可以直接去写 SQL。

即便你使用的是 MS SQL Server，你也不能去调用它特有的 SQL 特性。

Asp.Net 刚出来的时候，微软曾经鼓吹过一个叫 N-Tiers 的架构：http://msdn.microsoft.com/en-us/library/ms973279.aspx / http://msdn.microsoft.com/en-us/library/bb384398.aspx 。

我曾经以为这是王道，直到我膝盖中了一箭 …… 呃，不，直到我看了 Joel Spolsky 写的 The Law of Leaky Abstractions：http://www.joelonsoftware.com/articles/LeakyAbstractions.html

理想很丰满，现实很骨感。

ORM 工具再怎么封装都好，底层用了数据库，就是用了数据库。

开发者必然需要了解数据库的特性，能否直接调用数据库的特性，是一个选择。

是否要彻底对上层屏蔽掉数据库的存在，也是一个选择。

N-tiers 架构推荐一层又一层的封装，如果错误使用，把选择当成教条，是会有噩梦的。

Python 是一门很有趣的语言，它支持继承，能实现 OO，但是缺乏 encapsulation 的语言支持。

Python 根本就没有 public / private 这样的关键字，然后呢？

然后可以回过头再去看："这样的场景本来就不应该出现，这是对象设计一开始没有做好造成的，错误的应该设成 Public 的属性设成了 Private"。

这句话，这话说得对嘛？

作业：

1. N-tiers 架构的噩梦场景是？

2. 什么系统 / 场景需要充分使用特定数据库的特性？ ■

Qcon 2012开发者关注云计算与敏捷

2012 年 4 月 18 日，Qcon 2012 北京站的活动在北京京仪大酒店举行。本次大会将从大数据处理、移动开发技术、网站架构、敏捷开发和项目管理方面展开议题。

Qcon 大会的参会者都是企业技术人员，大家不光来 Qcon 学习提高，同时也带来了工作中遇到的问题。期望透过大会讲师的点拨，让这些疑问迎刃而解。

在乐视网数字家庭产品部总监于刚看来，云计算技术确实能给网站在数据处理和成本控制方面带来长足的改进。而透过相关讲师的意见中，我们知道目前很多开发者在选择云平台的过程中还存在盲区。比如过于迷信硬件设备的规模，超需求投入等问题。做云计算还是循序渐进的方式比较负责任，当用户出现问题时也能及时解决。

目前全国都在大张旗鼓的新建云计算数据中心，这种模式很像当年的修路铺桥的 GDP 模式。在政府资金的鼓励下，IDC 遍地开花。软性的服务是不是能跟得上这种建设速度，还是个大大的问号。

国内的开发团队面对日益增长的需求，已经有更多的开发团队尝试敏捷开发。但来自军队背景的 X 经理说他们也在考虑敏捷开发，让 51CTO 记者也有所惊讶。军队软件开发历来要求高安全性，开发周期自然不是最重要的考虑项。不过 X 告诉我们，国外军队的科技项目是始终领先于民用技术的，比如互联网等都是军队背景的产物。但在中国，军队项目是萌芽于民用项目，然后再进行拔高。这样他们的公司必然要先做一些民用项目，再移植到军队中，这里就很需要敏捷开发的方法论了。

对比以往的 Qcon，这次的北京站活动，外国听众的人数有了长足的增长。他们是在中国上班的软件开发人员，这个现象值得引起我们的注意。

51CTO 报导节选：

肖伟详解百度个人云对移动开发的支持

肖伟指出，目前国内外有很多的公司都有不错的产品架构来支持云计算这个理念，然而，大多数产品只是将传统的 IDC 建设，变成了一个软件化的产品，也就是给开发者提供机器和带宽，让他们的服务在这个云服务器上运行。

大众点评网王宏：从 .NET 迁移 Java 平台

大众点评网成立至今已有 9 年，王宏在大众点评网的平台迁移过程中面对了许多转型的问题也积累了宝贵的经验。他表示，“目前网站除 OA 系统仍然使用 .Net 平台外，其他的平台均已从 .NET 迁移到 Java 平台；.Net 升级快，但是 Bug 也不少，部分原有平台级的问题无法快速解决。”

开发应用的目标不是赚钱 而是创新

在 QCon2012 全球软件开发大会上，传奇产品工程师 Mike Lee 结合了自己八年的开发应用的经验，向参会者分享开发靠谱应用的一些方法

欢迎各位网友访问



http://developer.51cto.com/art/201204/332237.htm ■

编者按

这传说大象在临近生命终点之时，会独自离开象群前往族群墓地以寻求最后的安息。那么程序员们呢？他们在步入不惑之年后，又该何去何从？

编程行业换代快，老前辈们今何在？

我们都知道，软件开发工作是年轻人们的游戏。在头脑灵活、精力充沛的年轻程序员薪酬猛涨、职位激升的同时，老程序员们无疑面临着更大的就业压力。这一切似乎已经成为不争的事实。

加利福尼亚大学戴维斯分校的计算机科学教授 Norman Matloff 最近刚刚发表了一篇社论，他在文中将软件工程描述为一个“职业生涯的死胡同”。“许多程序员都发现在三十五岁后，自己在行业内的竞争力开始明显下滑，” Matloff 写道。

幸亏这不是什么电台新闻，不然我一定会把他的话录下来。35 岁？为什么要以 35 岁来界定程序员是否已经步入职业生涯的末期？

“统计数据显示，大多数软件开发人员会在 40 岁左右离开这一领域，” Matloff 补充道，读到这里，我不禁怒向心头起。“大多数”程序员？怎么个大多数？统计对象是谁，我咋不知道呢？他们转行之后又在从事什么行业？（顺带一提，Matloff 拒绝披露他的统计数据从何而来）

就算他所说的情况属实，那么这些老程序员们干嘛去了？那些为编程行业贡献了宝贵青春的前辈们仅仅由于年龄和精力问题，就被我们这帮新生一代直接踢出团队，这可能吗？他们接下来走向哪里，是跑去经营房地产，还是帮商场处理货品清单？说实话，我对这个结论深表怀疑。

天真要塌下来了吗？

我并不打算以高风亮节的姿态高调宣称软件开发行业中不存在什么年龄歧视。事实上科技领域中人才换代确实比其它行业更加频繁。不过我必须得说，Matloff 的这篇言论实在过于令人震惊，我的直觉让自己本能地对这种观点表示抵触。

首先，我要谈谈自己身边的情况：我的人际圈子可以说相当之大，但由于自己年龄的关系，这个圈子中的朋友和同事们基本都已经人到中年——其中不乏许多软件开发领域的从业者。但令我感到奇怪的是，Matloff 所表达的观点在他们当中丝毫没有体现。难道我所接触到的人群碰巧没被“统计数据”考虑在内？

除了统计数据，Matloff 的耸人听闻的言论还选择了一些顶级业界巨头的事例。例如他援引前英特尔公司 CEO Craig Barrett 的言论，称“软件及硬件工程师们的生命周期只有数年。”此外，他还提到了 Mark Zuckerberg……算啦，咱们直接无视这家伙就好了。

甚至连 Gordon Moore 在即兴发言中无意提到的内容都被他奉为“金科玉律”，这里之所以直接引用 Craig Barrett 的原话，是因为他确实曾经抱持过跟 Matloff 一样的观点。但断章取义无法代替严谨统计，因此我还是要问：程序员们都去了哪里？虽然我无法进行大规模取证，但在这里我打算说说自己的看法。

稳健路线

首先，有一些程序员肯定没有抛弃自己所熟

悉的技能。在某些领域、行业、工具使用或者企业中，他们以高度专业化的姿态集中处理专项事务，并坚持通过自己最擅长的知识开拓这片最为自己所了解的市场。这样的程序员群体必然会成为各大型科技企业中的“杰出工程师”，不过在 Matloff 眼中他们当然应该被划在调查对象之外，因此我们姑且不做进一步讨论。

其它程序员们则不可避免地通过晋升走向管理岗位。我知道，大家对此半信半疑。我猜各位一定会说：“在软件开发行业积累下来的多年经验并不能帮你成为优秀的管理人才。”但大家知道吗，指导及培训方面的工作正是老程序员们最为擅长的领域。

不过这还不是我要表达的重点。所谓晋升，最直观的感受在于改变我们的职称头衔。向大家展示自己的新头衔是让别人了解到自己岗位提升的最佳方式。因此在面对调查采访时，老程序员们往往会在职位角色一栏中填写“项目经理”，而非过去的“软件开发人员”——即使他们仍然会直接参与到软件设计、规范、开发、测试及部署的每一个步骤，头衔的变更仍然会成为他们引以自傲并乐于展现的业绩证明。

及时适当的晋升机制是留住员工的有效方法。但无论大家是否相信，走向软件开发团队的管理岗位并不意味着我们会彻底挥手告别软件工程师的角色。虽然看起来自己已经成为管理层中的一员，但所负责的工作内容还是要看企业的具体编制设定。

低调处事、开拓新路

另一些开发人员则选择留在开发领域，却辞去原有工作。他们开始在新兴企业中找寻自己的价值，在那里他们的头衔很可能变为主管或者 CTO。而企业的领导者往往不会被纳入就业情况调查名单——因此，他们也没能被反映在统计数据当中。

就业情况调查还很可能漏掉独立承包人。对于经验丰富的软件开发人员而言，为其它企业或团队提供咨询服务也是实现个人价值的上佳选择。他们厚重的积淀与过人的阅历不仅引导着新项目走向成功，也能让自己获得极大的经济回报。

然而，既然我们不从事全职工作，那么调查人员肯定很难把握大家要花费多长时间才找到新的全职岗位。相信各位一定注意到了，国家所统计的失业数字中永远不会包含“沮丧的群众”一项，因为这些已经不打算尝试寻找工作的群体根本没法反映在结论中。同样，独立承包人也很难体现在统计数据当中。

顺着这条思路，我们再看看 Matloff 口中的统计数据是怎么回事。他声称，根据某调查，“四十岁以上的程序员在失业后，寻找新工作所耗费的时间比平均水平高出 23.4%。”这个结果很有意思，但研究机构根本就没有深究其中的原因——我并不质疑他们的智力水平，可能他们在发现真实情况后故意不说，因为老程序员孤苦无依之类的结论才具有话题性。

值得一提的是，老员工对于新工作的要求及顾虑也比较多。他们也许搞砸过某些项目、选错了工作岗位，并习惯于某种特定的生活水平。我想对于一个多年以来一直舒适生活的程序员来说，他肯定不会去跟刚毕业的毛头小子们争夺某个职位。（本文未完，更多请阅读原文）■

http://developer.51cto.com/art/201205/335141.htm

谷歌I/O 2012开发人员该关注什么

谷歌年度开发者大会门票在二十分钟内即告售罄，到底是哪些内容吸引了全世界技术爱好者们的目光？

什么样的开发者盛会能在短短二十分钟之内就将五千五百多个入场名额销售一空？这近乎天方夜谭的状况却在谷歌 I/O 2012 年度大会中变为现实。门票销售之热烈一举打破了该大会在去年所创造的一小时售罄之纪录，旧日的神话如今再度被改写。

其实只要略加分析，我们就不难发现谷歌 I/O 大会销售之火爆与现场赠品的丰厚不无关联。尽管门票本身价格不菲，但每位与会者都会依照惯例得到一份神秘的礼品袋。就在去年，谷歌豪爽地在袋中向广大参与者送出了一台三星 Galaxy 平板设备以及一台 Chromebook。而前年大家得到的礼物则是一台 Android 智能手机。从这个角度来看，门票本身所代表的参与资格几乎可以算是免费的。

然而成也萧何是败也萧何，丰厚赠品的背后却也带来了沸腾的民怨。人们纷纷吐槽称大部分与会者都是冲着礼物而来，大会所吸引到的“货真价实”的开发人员其实非常有限。不少民众甚至指责日益上扬的门票售价与愈发有限的门票供应已经使谷歌大会渐渐失去了让全世界开发者欢聚一堂的初衷，而变得越来越像是面向企业的商业嘉年华。不过我倒不这么看。在过去的几年中，谷歌 I/O 大会已经吸引到了来自各个领域的开发人士。随着谷歌在开发生态系统方面的不断努力，日益多元化的内容令不同专业、不同侧重的顶尖开发人才聚集在这一盛会当中。无论是 Web 还是 Android，总有适合我们的表现平台。

这次大会直到六月底才会正式举行，不过大家对它的关注从现在开始就已经处于持续升温状态。今年的极客狂欢将带来哪些振奋人心的内容？我先抛砖引玉，提出以下几项期望。

Android 继续勇猛精进

谷歌潜心打造的这款移动操作系统自然是大会的重头戏之一。问题在于，家搜索业巨头会不会在活动中发布该系统的下一个新版本。根据小道消息，代号“Jelly Bean（软糖豆）”的 Android 5.0 将在此次大会上正式亮相。

这一消息不禁令人担忧。去年的大会将宣传重点放在 Android 3.0 “蜂窝”之上，而谷歌又在前一段时间发布了代号“冰淇淋三明治（简称 ICS）”的 Android 新版本。与频繁升级形成鲜明对比的是，当前移动运营商们只为 1% 左右的 Android 智能手机配备了 ICS 系统。就连去年作为谷歌 I/O 大会赠品放出的 Android 旗舰级平板设备都没有使用 Android 4.0，这种情况令我们不禁怀疑，如果六月份真的再来个什么 Android 系统新版本，ICS 岂不是等于打了圈酱油就得解甲归田？

尽管如此，谷歌的既定安排还是会照常进行。如果智能手机与平板设备制造商们真的跳过 ICS 直接采用“软糖豆”，至少新产品上的 Android

谷歌 I/O 2012 大会，开发人员该关注哪些内容？ II

碎片化问题会大大得到缓解。

除了操作系统，真正吸引眼球的要数 Android 硬件领域。当下的 Android 智能手机市场在多样性方面显然无法令人满意，虽然品牌与型号堪称“乱花渐欲迷人眼”，但各产品在规格及配置方面似乎颇为雷同。而且设备制造商们对于谷歌所热衷的尖端技术似乎兴趣不大，就拿近场通信（简称 NFC）技术来说吧，谷歌一片热诚，厂商却始终虚与委蛇。不出意料的话，这次谷歌 I/O 大会应该会拿出新的 Nexus 智能手机机型，作为 Android 硬件市场的一次基准式提升（同时也是为了向苹果示威）。在 Android 平板方面基本也会有类似的举动，算是为这片市场注入一剂强心针。时至今日，Android 平板在与 iPad 系列产品的竞争中基本属于一败涂地。但谷歌绝不会坐视这种势头继续发展。一款教科书级新平板，屏幕大幅改善、多媒体功能显著提升，我想这样的还击算是相当给力了。

向 Web 平台进军

从技术角度来讲，HTML 5 规范还未彻底完工。但以谷歌、Mozilla 及其它多家行业带头人为首的业界已经将 HTML 5 视为未来一段时间内 Web 领域的新标准。目前他们将此称为“Web 平台”，相信在此次谷歌 I/O 大会上我们会频繁听到这一名词。

在上周的 Web 组件项目专栏中，我刚刚写了篇文章介绍 Web 平台上的开发事宜。希望在今年的谷歌 I/O 大会上，我们能看到这些新技术的相关内容。除了之前提到的项目之外，Web 技术方面的新消息可谓层出不穷；而谷歌公司在此类开发活动中始终扮演着举足轻重的主要角色。最突出的例子就是 Web Intent，这是一款用于实现多款 Web 应用程序间互操作性的框架。谷歌 I/O 大会的召开相信会为这些即将付诸实践的 Web 技术带来良好的推动作用。

当然，谷歌打造的 Chrome 浏览器想必也不甘落后，更新消息应该将于大会中曝出。在六月份大会召开的时候，我希望谷歌倾力铸就的 Chrome Canary 版——一款领先的浏览器分支版本——能够在保持自身优势的前提下融入主流市场。而其中最令人心动的，要数高达 25% 的 JavaScript 性能提升。有了这样的进化，Chrome 必然会在迈向浏览器王者道路上走得更加稳健。

认真对待娱乐领域

相对于以往关注度的低下，本次谷歌 I/O 大会可能会刻意加强游戏开发领域的关注力度。根据当前的总体形势，娱乐已经成为谷歌公司成长速度最快的重点领域之一。不知大家注意到没有，谷歌公司以 Android Market 为原型，专门针对娱乐软件推出了一套名为 Google Play 的新型软件商店，意在将电子书与多媒体等内容与传统应用紧密结合起来。

长久以来，游戏一直占据着 Android 软件下载榜中的头把交椅。如今谷歌则希望以游戏为契机，向广大用户展示 Web 平台的强大潜能。请大家注意，谷歌 I/O 大会的网站本身就由基于浏览器的解谜游戏所表现，这无疑意味着广大开发人员应该能够在大会上了解到许多游戏制作中必要的浏览器功能。其中不仅包括时下话题率极高的 HTML 5 特性，还将包括 Chrom 浏览器所特有的多种功能。（本文未完，请阅读原文）■

http://developer.51cto.com/art/201204/328737.htm

想要快速持续交付 何以如此困难?

在开源领域,现在 Chrome、Firefox 等项目基本都保持六周一个新版本,管他功能实现完了没,先上了再说。几个著名的 Linux 发行版本,则保持六个月一版的发布周期,但是测试版的周期也都维持在一个月的长度。感觉在互联网行业,敏捷开发理念的进展、持续交付的状态的确比两年前的企业级开发领域要好很多了,但也有很多不尽如人意的无奈。

互联网时代的应用开发讲究快速迭代,就算没有一天发布个三版,怎么说也得两天发一版才好意思出门跟人打招呼。而移动应用发布的周期,一般也在一个月以内。

在开源领域,现在 Chrome、Firefox 等项目基本都保持六周一个新版本,管他功能实现完了没,先上了再说。几个著名的 Linux 发行版本,则保持六个月一版的发布周期,但是测试版的周期也都维持在一个月的长度。另外,大部分成熟的开源项目都有 Nightly Build,即一天一个新版本。

这里面的区别当然也很容易看出来,互联网应用不用下载,用户上去就能用,改动越频繁、每次的改动越小,则越容易收集反馈,对产品更有利;而像是操作系统这种大块头,下载一个镜像就要几个小时,安装升级再几个小时,肯定不能让用户频繁的折腾。

不过无论是哪种产品,业内当前对“应用交付”这方面已经有了统一的观点,那就是无论你什么时候去看产品开发的最新进展,它都应该是可用的,最好能处在一个可随时上线的状态。

笔者上个月在百度开发者大会上遇到了百度高级架构师路宁(@路宁同学),跟其他几位开发者一起聊了一些敏捷开发和持续交付方面的话题。感觉在互联网行业,敏捷开发理念的进展、持续交付的状态的确比两年前的企业级开发领域要好很多了,但也有很多不尽如人意的无奈。

“一开始接触敏捷开发的开发者,都很不习惯那种每天工作透明化的状态,压力很大。”

“我的团队都是些新手,只有一年经验的或者完全就是毕业生的,上来让他们做预估,基本都是不准。一个功能预估做两天,结果做了一周,经常都是这样。”

“好不容易这个人成长起来了吧,别的企业待遇稍微好点就把人挖走了。而领导也不会太多考虑你的难处,只想用廉价劳动力,就再招新人进来,如此反复……”

这些无奈的声音来自一位从业六年的开发者。王先生现在在一家电子政务方面的企业做产品研发经理,负责整个研发团队的工作与培养。

“有些小孩进来吧,你也不能说他不努力,但总感觉心不甘情不愿的。”王先生讲到自己的工作时,说到自己曾经有段时间每天工作 18 个小时,吃饭什么的全抛掉,只有 2、3 个小时用来睡觉。当然,他也表示不打算再这样拼命了。“我们给毕业生的待遇也不算高,而敏捷开发的那种透明,虽然能够让他们快速成长,但确实造成很大的压力。”

敏捷开发的理想状态,是一个团队在项目进

想要快速持续交付 何以如此困难？ II

行过程中能够不断成长，对项目越来越熟练；每次项目都是一次练兵，而团队的整体能力则会越来越强，在敏捷开发的流程当中也会带来越来越高的生产效率。然而，理想是丰满的，现实是骨干的。对于国内的大多数技术团队而言，不稳定可以说是一种常态。老板希望找廉价的工作力，还要进来就能用，进来的新人无论在能力还是忠诚度方面都不能尽如人意，在这种环境下想要做到准确的预估工时，想要让他们坚持透明化的运作，是非常困难的事情。最终，还是团队的人员质量决定了持续交付的效率和质量。

一方面是团队的参与者。“现在我招聘的时候，都会非常看中这个程序员的心态是否开放。他如果是那种特别以自我为中心的，听不进别人的意见，也不愿意沟通的，这种我就会比较担心一些。”路宁介绍说他自己带过一些团队，有的人就很不愿意做 Pair Swapping，因为不愿意让别人看到自己的工作进度，这样的话就比较难展开。所以，开放的心态是非常重要的。

另一方面，团队的管理者也很重要。路宁表示有的管理者会滥用敏捷开发流程当中的透明性，反而对团队的发展不利。“比如，透明化之前的工作可能有张有弛的，这对于开发者来说是一种常态；但透明之后管理者可能看到，你原来这么快就把这个功能做完了，那我要给你加量，反而使得开发者无法得到充分的喘息。”

对于敏捷开发团队的规模，路宁也提供了一些建议。一般敏捷团队从 6、7、8、9 个人到 20 个人都是可以的，不过也有的团队到 10 个人之后就搞不到一起去了。一般来说，6 ~ 12 个人的团队来完成持续交付会比较理想。

51CTO 精品杂志推荐《Linux 运维趋势》

《Linux 运维趋势》是由 51CTO 系统频道策划、针对 Linux/Unix 系统运维人员的一份电子杂志，内容从基础的技巧心得、实际操作案例到中、高端的运维技术趋势与理念等均有覆盖。



本期为 2012 年 5 月号，总第 19 期，主题为运维安全。

本杂志长期处于探索期，需要更多来自大家的意见与参与。

如果您对这份电子杂志感兴趣，那就请您下载阅读；想要帮助我们做的更好，请进入我们的 Linux 运维趋势技术圈留下您的宝贵意见建议。

读者讨论组：http://g.51cto.com/linuxops/

邮件订阅入口：

http://os.51cto.com/art/201011/233915.htm

投稿信箱：yangsai#51cto.com

发布周期：每个月的第二个星期五

往期《Linux 运维趋势》下载汇总页：

http://down.51cto.com/zt/71

下载地址（51CTO 下载频道）：

http://down.51cto.com/data/329527

■

编者按

本系列文章将围绕 SQL Server Profiler 进行，今天将是系列文章的第一篇——原理与相关概念介绍

全面掌握SQL Server Profiler系列:1

通过本篇文章的越多，朋友们会清楚如下的问题：

1. 理解 SQL Profiler 是如何工作的

2. 对 Profiler 的一些术语和概念有一个比较清楚的认识，例如事件，数据列，过滤器等

3. 理解如何配置权限启动 Profiler

4. 可以创建基本的 Profiler 跟踪，并且保存。

在开始学习任何一门技能之前，我们需要知道：为什么要学习这门技能，学习完了之后，可以给我们带来什么好处？ 毕竟人的精力有限，不可能随随便便学习一门对自己毫无用处的知识。

可以这样说：区分一个有经验的 DBA 与一个新手 DBA 的区别就在于：有经验的 DBA 知道在核实充分利用何种工具可以快速的将问题搞定（其实我本身也是一个开发人员，通俗点说，就是写应用代码的，不算是真正 DBA，因为很多的时候，在项目中，没有 DBA，所以不得不扮演很多的角色，所以学习了不少的技能）。

如果熟练的掌握了 SQL Profiler 之后，有经验的 DBA 就可以对数据库的运行状况进行跟踪，并且利用跟踪的数据进行问题的分析与性能的诊断与优化，有时候，甚至可以再问题还有发生之前发现潜在的问题。我们把掌握 SQL Profiler 的好处列下来：

1. 可以监视 SQL Server 内部发生的各种活动：例如发生在数据库引擎中的活动，发生在 Analysis Services 及 Integration Services 中的活动。

2. 监控并且确认某些查询，事务相关的性能问题等。

3. 可以对执行计划进行分析。

4. 可以监视并且确认 SQL Server 内部的发生的一些错误和警告。

5. 可以捕获数据库发生的数据，并且还可以对数据库的压力进行重放。

6. 可以对数据库中用户活动进行审计跟踪。

7. 通过多种方式对数据进行分析与跟踪。

8. 可以将跟踪的数据与系统的性能计数器关联起来，从而更加全面的检查发生的问题。

9. 可以将跟踪的数据以 DTA 来进行分析

10. 还可以对 SQL Profiler 进行编程，可以不使用它的 GUI 工具。

说了这么多，似乎已经把 SQL Profiler 说的天花乱坠了！ 其实 SQL Profiler 也不是万能的，也只是在 SQL Server 诊断与分析中常常使用的一种工具而已，至于决定到底是否在使用它，还是取决于 DBA 或者技术人员本身，通过本系列的学习，相信大家会有一个比较清楚的认识。

首先，我们来简要看看 SQL Server Profiler 的工作原理

其实 SQL Server Profiler 这个工具是 SQL Trace 的一个 GUI 的版本，而 SQL Trace 是一组脚本，自 SQL Server 2000 就已经有了的，这一组

全面掌握 SQL Server Profiler 系列 1: 原理与相关概念 II

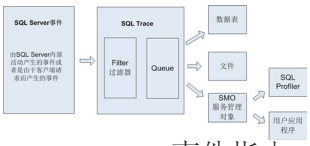
脚本捕获和跟踪 SQL Server 内部发生一些事件，并且将跟踪的数据保存以便后续使用。

从本质上说，SQL Trace 就是一个非常简单的工具，它的作用就是捕获客户端应用与 SQL Server 之前的通信的信息。它扮演中一个非常特殊的网络嗅探器的功能，用来捕获

与 SQL Server 相关的网络活动，同时也允许我们查看客户端应用的请求触发了 SQL Server 的哪些事件。

另外，与我们熟悉的网络嗅探器不同的是：SQL Trace 只是捕获发生在 SQL Server 内的相关的事件和活动，而不会获取网络上面传输的数据。

为了使得大家对 SQL Trace 有一个更加清楚的认识，我们来看看下面的一个图 1-1：



1.SQL Server 事件指由 SQL Server 本身内部触发的活动，或者由客户端与 SQL Server 进行连接时在 SQL Server 内部触发的活动。事件包含很多不同种类，我们将会在后续文章中详解介绍。

2. 因为事件发生在 SQL Server 的内部，SQL Trace 的任务就是根据 Filter 来捕获自己感兴趣的事件。

3. 一旦事件被 SQL Trace 捕获，那么发生事件与相关的事件信息就会被保存在内存队列中。

4.SMO(Server Management Objects)，是可以直接的访问 SQL Server 的对象模型。其实，SQL Trace 对于我们而言，就可以将之看成是一个黑盒，我们能做的就是通过一些方式和工具去与这个黑盒交互，SQL Profiler 和系统的存储过程就是这样的工具。

为了后续的理解方便，也为了保证我们的理解一致，我们下面介绍一些常见的概念和术语。

事件

相信熟悉微软技术的朋友对“事件”这个概念已经不陌生了，简而言之就是发生的活动，例如一个存储过程的执行，就是一个活动，也是发生了一个事件。SQL Profiler 允许我们捕获大约 170 多种不同的与 SQL Server 有关的时间。

另外，在 SQL Profiler 中，使用 Event Category 这个概念，对不同的事件进行了一个大致的分类，使得为我们的监控更加的方便，例如所有与存储过程相关的事件都被划分在一个分类中。

数据列

这个概念理解起来非常简单，简而言之就是描述事件发生的一些信息。这个概念和我们在 .NET 编程中的事件信息的概念是一样的。如图 1-2 所示，可以看到事件发生的时候，包含了一些实事件信息，即数据列。



图 1-2 中列出了很多事件的数据列，例如，对于 RPC:Completed 事件而言，包含了 TextData，ApplicationName 等。同时，还可以看出，不是所有事件都有相同的数据列。

过滤器

顾名思义，过滤器主要是 SQL Profiler 用来对发生的事件进行过滤的，只捕获感兴趣的事件。

本文未完，更多内容请参考原文 ■

http://database.51cto.com/art/201205/334648.htm

别怕！我们还有HTML 5和CSS3

HTML5 和 CSS3 热潮正横扫网络，从事互联网行业，特别是前端工作者，大家或多或少都有了解、学习和使用，HTML5 和 CSS3 包含丰富的技术内容，在每周的技术分享交流会中，页面构建工程师 Sunshine 珍　同学，从三个特别的角度出发，配合丰富的实例，概括了 HTML5 与 CSS3 对页面工作带来的息息相关的变化和精彩。

一、我们担心苦恼的那么事儿

1.1、对于页面构建者来说：

表现层会遇到一些难题，欣赏绚丽的 UI 设计稿的同时，也在苦恼怎样用现用的技术力量去实现圆角，阴影，高光，渐变，透明等各种表现效果。为了精准 UI 还原度，我们会不惜花费更多字节数切大图或者添加一些多余的空标签。

结构层同样也会遇到麻烦。首先看页面首行代码 doctype，包含了，冗长而复杂，从事多年开发的你能记住这段话吗？

接着看满屏几乎都是毫无语义的 div 标签，为了实现复杂的设计效果，我们不得不使用多个层级嵌套，造成代码冗余。

再看看页面中一些交互动画效果，基本都是 JavaScript 和 Flash 开发人员所实现，页面构建在现有技术上只能有心而无力。



1.2、对于 js 开发人员来说：

一个简单的交互效果，一个又一个表单验证，需要使用一大段 JavaScript 代码去实现，为了兼容各个浏览器，不得不多写数行代码。

1.3、对于 Flash 开发人员来说：

Flash 开发人员担心的是用户没有安装必要的插件，担心插件被禁用或者屏蔽，像现在 apple 的 ipad 就已经不支持 flash 插件了。

以上的担心与苦恼你曾有过吗？如果有，别怕，HTML5 和 CSS3 来了，已经来了！

二、我们开心盼来的那么事儿

HTML5 和 CSS3 的到来，让我们网页开发者可以做的更多，更好！

CSS3 带来了圆角，半透明，阴影，渐变，多背景图等新的特征，轻松实现了设计稿中常见的图层样式，用简洁的代码替代图片，代替了多余的空标签。

CSS3 带来的媒体查询可以为不同的显示设备定义相匹配的样式，灵活实现了智能的流体布局，CSS3 还为我们带来了强大的选择器以及变形动画。

下面让我们一起来看 看实际工作中的具体应用吧 .

实例一

类似实例一的设计稿，我们常见的做法是在无序列表 li 里为不同小图标识写上不同的 class 类名，或者单独为每个小图标写上空标签定义样式，这样带来的问题之一是 HTML 结构不干净简洁，二是使后续的开发工程师多了一层判断。

而利用 CSS3 强大的选择器就能轻松解决这个问题，E[foo^=" bar"] 表示的是选择匹配 E 的元素，且该元素定义了 foo 属性，foo 属性值包含前缀为 bar 的子字符串 .

CSS3 包含了很多新的属性特征，动画效果尤为受大家喜爱。实例二是为这次分享做的一个学习转动盘。用 JavaScript 和 CSS3 去控制转盘的指针转动效果。



实例二

该实例重点在于 animation-play-state 属性，paused 表示的是暂停。

Animation 属性包含了动画名称 Animation-name，动画时间 Animation-duration，动画播放方式 Animation-timing-function，动画开始播放时间 Animation-delay，动画播放次数 Animation-iteration-count，动画播放方向 Animation-direction 等等。

领阅了 CSS3 带来的价值，那么 HTML5 给我们带来了哪些盼来的事儿呢？

HTML5 的口号是简单至上，尽可能地简化：简化了 doctype，如；简化了字符集声明；新增了更多语义化标记 header、footer、section、article、aside、nav；新增了很多标识元素如 canvas、audio、video，配合简单而又强大的 HTML5 API，让浏览器发挥它原生的能力，来替代复杂的 javascript 代码。

实例三

用 Canvas 画的一个 weibo logo 图标。这里用到了两个知识重点：

其一，context.lineCap=[value]，线帽风格有三种：对接（butt）默认值，圆形（round）方形（square）；

其二，arc(x, y, radius,startAngle, endAngle, anticlockwise)，x,y 指定绘制的圆弧的圆心下 xy 轴坐标，radius 是圆的半径，startAngle 和 endAngle 指定了起始弧和结束弧，anticlockwise 指定是否使用逆时针方向绘图，逆时针方向（TRUE），顺时针方向（FALSE）。

实例四

用 Audio 标签做的一个音乐播放器，用 HTML5 API 提供的属性去控制该控件的播放 play()，暂停 pause()，进度条，当前时间 currentTime，音量 volume，歌曲选择且封面和曲名相应变化等效果。

■未完，查看网络原文：

http://developer.51cto.com/art/201205/335355.htm

JavaScript写法你更偏向哪个

作者 / 穿山甲

思维导图



介绍

老是在写 js，你平时是怎么写你的 js 呢？更喜欢哪种方式呢？

今天就把常见的跟大家唠叨唠叨。——年纪大了，就这一爱好（平时也喜欢说胡话），别闲我烦！




```
var userData = {
    userData:null,
    code:'010',
    name:location.hostname,
    getName:function(){
        return this.name;
    },
    getCode:getcode
}

// 定义对象中的方法
function getcode(){
    return this.code;
}

// 添加新方法
userData.setUserData = function(userData){
    this.userData = userData;
}
userData.setUserData('Chuanshanjia\'s UserData');

// 添加新类（模块）
userData.email = {
    getCode:function(){
                //return this.code;
        return userData.code;
    },
}
console.log('email\'s code:' + userData.email.getCode());
```

需要关注一下，也是我们经常要使用的方法。

直接初始化法

优点：1、userData 本身就是对象实例。上来就实例化一个类出来，也不怕浏览器受不了；2、代码紧凑；3、编程效率高。

缺点：1、代码的重用性比较差；2、不符合面向对象的编程思路。——设计模式这种玩意，经常是在类上，弄来弄去。这种直接在对象上使用，够呛！

总结：应避免使用该方法创建自定义对象。

构造函数法

```
function userData(){
    this.userData = null;
    this.code = '010';
    this.name = location.hostname;
    this.getName = function(){
        return this.name;
    }
    this.getCode = getcode;
}
// 定义对象中的方法
function getcode(){
    return this.code;
}
// 添加新方法
userData.prototype.setUserData = function(userData){
    this.userData = userData;
}
// 添加新类（模块）
userData.prototype.email = {
    getCode:function(){
                //return this.code;
                return user.code;
            },
}
// 静态方法或方法
userData._static = function(){
    return '_static';
}
console.log(userData._static());
var user = new userData();
user.setUserData('Chuanshanjia\'s UserData');
console.log(user.userData);
console.log('email\'s code:' + user.email.getCode());
```

不用prototype，直接在后面使用相当于一个静态方法。

使用方法：需要使用“new 和 构造函数”创建实例。

优点：只有 new 的时候，才分配内存。如果不 new，浏览器一辈子都不会给你内存资源的！

Object 表达式法

类似于第一种”直接初始化法“。——其实是一种变种。

■未完，查看网络原文：

http://developer.51cto.com/art/201205/334153.htm

开发HTML 5跨平台游戏相关经验

我们刚成立 Gamezee 的时候，行业中许多人都对 HTML 5 抱有成见，就算是一些对其较为乐观的人士也只是说，HTML 5 将是未来发展潮流，但目前还不够成熟和稳定，无法制作出当前主导社交游戏领域的 Ville 类型 Flash 游戏。



我们刚成立 Gamezee 的时候，行业中许多人都对 HTML 5 抱有成见，就算是一些对其较为乐观的人士也只是说，HTML5 将是未来发展潮流，但目前还不够成熟和稳定，无法制作出当前主导社交游戏领域的 Ville 类型 Flash 游戏。

当时我们准备使用 HTML5 制作一款可同时运行于 PC 端的 Facebook，以及所有 iOS 设备和 Android 手机平台移动网络的大型等视距游戏。

这款出自单个团队之手，并且基于 HTML5/node.JS 的游戏《Skyscraper City》仅运行一个代码库，它是我们这家工作室的开山之作。

《Skyscraper City》这款城建游戏现在已经入驻 Facebook 及 iOS、Android 手机平台，结合了社交建设玩法，含有乐高积木风格的元素。玩家可以堆叠任何一种城市单位以建设高塔，或者装饰建筑等。

开发这款游戏并非易事，以下是我们所总结的一些经验：

1）移动设备的玩家数量多于我们的预期。

我们想制作跨平台 HTML 5 游戏的原因之一就是，我们自己也是游戏玩家。我玩过《CityVille》，并希望外出时也能在手机上玩到这款游戏。Zynga 虽然推出移动版《CityVille Hometown》，我也可以在 iPhone 或 Android 手机下载这款游戏，但它与原版《CityVille》并不相同，它是另一座城市。而我们的游戏却能提供真正的跨平台体验，你可以在 PC 电脑、平板电脑、移动设备上玩游戏，并且实现游戏进程在所有设备上的同步。

我们认为多数用户的想法也是如此——他们多数时候在电脑上玩游戏，然后在其他场所时就拿出平板电脑或手机继续体验游戏。但在 iPid Touch 或 iPad 上玩游戏的用户要多于在 Mac 平台，而 Android 手机游戏玩家又会多于这些苹果游戏用户。

如果我们一开始就清楚这一点，就会花更多时间针对移动设备而非 Mac 浏览器优化游戏体验。虽然我们设计之初就考虑到移动设备，但我们却只是将其视为次级游戏环境。我们建议其他开发者将优质的手机游戏体验作为首要考虑目标。

2）必须进行多次试验。

我们一开始就有使用 Canvas 渲染游戏的念头。这在桌面电脑平台上十分管用——完全不存在运行效果的问题。但在 iPhone 3GS 平台，运行效果很糟糕（我们制作游戏时的每秒渲染画面不足 5 帧，但 iOS 5.0 发布之后情况大有好转）。

所以我们只好接二连三地创建一个又一个渲染引擎，直到创建出目前使用的 DOM 渲染器为止（它适用于多数设备）。

开发 HTML5 跨平台游戏相关经验 II

这个 DOM 渲染器使用 CSS 动画，支持我们在 2D 图像上使用伪 3D CSS 变换以触发移动设备上的硬件加速功能（这可以略微提升运行效果）。

3）HTML 5 仍存在一些难以攻克的软肋。

HTML 5 上的音效仍然很有问题。游戏开发需考虑的是植入多种声音以响应游戏提示信息及玩家输入操作。但尽管我们进行了多次试验，仍然无法在多数移动设备上实现较理想的音效（游戏邦注：例如，无法让声音即时响应游戏内容或玩家操作），也难以让游戏同时播放响声和音乐。所以我们就选择在电脑版本的游戏中同时保留声音及音乐内容，但移动网页版本仅保留音乐。

在 DOM 上提升运行效果的一个类似做法是在 CSS 上进行伪 3D 变换。这可以在多数移动设备上触发硬件加速，使其获得比 Canvas 更出色的运行效果

出于某种原因，有些 Android 设备（例如三星 Galaxy S21）并不支持这种操作，所以你无法针对它们提升游戏运行性能。

4）在设计之初就要有跨平台理念。

我们希望手机版本与电脑网页版本的游戏体验更为相近，这意味着我们需要关闭手机版中的缩放功能，在网页版游戏中移除道具拾取的翻转效果，并想法在手机版游戏中植入翻转（我们的想法是通过一个按扭激活或使用手指按压目标道具来实现这一点）。

像《CityVille》这类游戏设计如果不进行重大调整，根本就不适用于手机平台，因为它一开始就并非瞄准移动设备。它的菜单太大太复杂了，游戏中的东西多得无法在小小的 iPhone 屏幕中呈现（游戏邦注：所以 Zynga 才需要针对 iOS、Android 发布独立版本的《CityVille Hometown》，而非添加一些元素直接将原版游戏移植到手机平台）。



5）不要采用大量堆叠操作。

我们游戏的一大特色就是上文提到的堆叠操作。在城市中创建一个巨型机器人或者建设一个由四根细柱支撑的倒金字塔确实很有趣，但在手机平台上的堆叠操作却问题百出。如果用户的手指不够纤细，有时候就难以准确选中他们想点击的物品。

另外，与 Flash 一样，如果你在屏幕上拖动的对象越多，渲染时间就会越长。玩家在这款游戏中一开始拥有 10 X 10 的城市方格，最高可以建设 11 层楼。

如果每一格都建设这么高的楼，那需要渲染的建筑内容就多了。

虽然堆叠很好玩，但对于扩建后的城市来说却十分不利于玩家操作，会延长游戏加载时间。

■

未完，查看网络原文：

http://developer.51cto.com/art/201205/333912.htm

JS大项目代码的多人合作处理方法

Js 开发在整个行业的现状主要的有两点

维护成本高

多人协作项目的困境

Js 项目维护成本过高

经历过 10 年的 ie 统治时期的浏览器世界一成不变，和 06 年开始 ajax 的兴起带动垂死的 javascript 的复兴，浏览器市场也开始翻天覆地的快速变化，排版引擎和 js 脚本引擎开始得到不断的优化，js 的执行效率不断的得到提高，但是……..

我们的 js 一直没变过 . 现在所有的主流 Web 浏览器都遵守 ECMA-262 第三版，即实现的是 JavaScript 1.5 版 时间 2000

那么，js 依然是当年的 js，浏览器和对 web 的需求早已经不复当年。当年那个为做表单验证而生的 js 语言，如今承担的是复杂的界面 ui 和交互处理，以前只需要写十几行或者几十行的 js 现在需要面对几千上 w 行的需求。

那么带来的问题就是十年来 js 语言特有的随意性导致的各路 js 程序员的 js 风格各异。

Js 的弱类型特性导致项目中经常会出现 jser 的各种奇技淫巧。

上面的两问题直接导致的是 js 程序交接维护上的困难。面对 js 风格清晰易懂的 jser 的代码还好，如果接手 js 风格怪异太个性的 jser 的代码，并且如果还没有注释，那么，看代码真心会看到你吐血

编写规范的随意性 ，编写风格的个性化，是 js 项目维护成本的很大一部分原因

多人协作项目的困境

其实看过前面的内容，我们大抵的了解到，js 是应很简单的需求而生的，甚至简单到是用来做简单的表单验证的。也就是说，js 代码，在很长一段时间内，都是由一个开发独立完成的，多人协作上会出现的问题

1，全局变量命名混乱冲突导致系统错误

2，代码集中少量文件导致开发效率降低

3，容易出现功能重复开发

JS 开发编写方式的演进

函数 :js 作为函数式语言与生带来的特性，jser 们通过在 window 下注册一个又一个的 function 相互调用来完成产品们的需求（全局变量命名极为混乱和不规范）

类 :jser 为大型 js 项目开发的尝试，2006 年，Ajax 兴起。JavaScript 绝地重生，越来越多的后端逻辑放到了前端。网页中的 JS 代码量急剧增加。这时写函数方式组织大量代码显得力不从心。有时调试一个小功能，从一个函数可能会跳到第 N 个函数去。这时 jser 们前仆后继，进行各样尝试，类出现了，Prototype 率先流行开来。用它组织代码，用 js 的灵活性模拟类的编程，每个类都是 Class.create 创建。又有 YUI、Ext 等重量级框架。虽然它们的创建类方式各不同，但它们的设计终极目标却都是要满足大量 JavaScript 代码的开发而努力。■未完，查看网络原文：

http://developer.51cto.com/art/201204/332096.htm

编者按

去年很多朋友私下或新浪微博上在总结自己的职业生涯与职业规划，也感觉到很纠结与彷徨，尤其技术人的职业生涯，随年龄增加，一些优势逐渐丧失。

阿里巴巴离职DBA 35岁总结的职业生涯

导读：

去年很多朋友私下或新浪微博上在总结自己的职业生涯与职业规划，也感觉到很纠结与彷徨，尤其技术人的职业生涯，随年龄增加，一些优势逐渐丧失。4 月 13 日数据库技术大会的主办方举行的晚宴上，也让本人支持一个大家感兴趣又贴合实际的话，话题是：三十岁后的 IT 人 如何抉择人生路，遗憾的是预订酒店饭桌是几个房间打通的，而不是一个大厅方式（总共：8 桌），大家无法围绕在一起，为此只是小范围讨论了下，今天看到曾经阿里巴巴数据库团队的海波兄弟总结分享，非常有意义，转载给大家共同欣赏！

十五年前，你或许还不懂爱情，看 Jack 和 Rose 执手相看泪眼，只是蒙胧的心痛。十五年后，你会和谁一起走进影院，更会和谁一起，走到生命终点。

十五年前，我还不太懂技术，凭兴趣玩着 C 语言。十五年后，当铁达尼的旋律再次响起，我会选择和哪门技术，一起走到我职业生涯的终点。

序言

--- 回望雨中的园区，更是回望来时的路。

此时天空飘着小雨，正如丝丝织雨细如愁。我刚刚离职，一路走出园区，回望了一眼雨中阿里巴巴的大楼，思绪随着微风，四处飘散。突然想起几年前我刚到杭州哪天，也是细雨微风。我走进城西的创业大厦，终于见识了大师 Biti 的风采，也从此开始我的阿里巴巴之旅。

杭州的特点，是隐匿在山青水秀之间，有很多雅致的主题饭店，团队第一次出外聚餐，就是在山间绕来绕去，最后到了一处青山环绕、绿水长流之处。Biti 在会议上就“什么是架构”进行了精彩的评论，高度已经超越普通技术人员，至今我仍记忆犹新。但不久之后，Biti 调至支付宝，真是遗憾。而这让我面临了空前巨大的压力，因为 Biti 一走，我变成了团队级别最高的 DBA。初来乍到，入职刚满一月，忽然成为阿里系一个子公司的“首席 DBA”，惊喜之余，感觉压力之大、常常透不过气来。不过时间总一晃而过，此时此刻，刚刚走出阿里，掏出手机，拍了一张照片，算作我在阿里的留念，更是我人生一段最重要经历的留念。



回望雨中的园区，更是回望来时之路……

阿里巴巴离职 DBA 35 岁总结的职业生涯 II

来时之路：硅谷，IT 传奇的开始。

---- 人可以依靠梦想活一段时间，但不会一直下去。

“坐落在纵横交错的高速公路之间的，是巨型的商业园区和简陋的三卧两浴的平房住宅，其间最高的建筑是变电站的铁塔和架着电话线的水泥杆子。真正的工作是员工们在简陋的、用屏风隔开的小办公室里，眼睛盯着电脑屏幕静静完成的。每个人都试图做一些前人没有做过的东西。”

这就是李彦宏在《硅谷商战》中谈到的硅谷的样子。

曾经有两个在一家公司供职的工程师，去找硅谷著名的投资家史蒂夫 · 朱维森，这位资深风险投资家对他们提出网上数据库系统并不感兴趣，他问：“你们还有其他主意吗？”

其中一个叫比尔 · 巴迪亚的工程师就说，他还想通过因特网来提供免费的电子邮件账户，从网上广告为获取收入。10 天之后，朱维森连商业计划都没要便掏出 30 万美元帮助他们成立了 Hotmail 公司。Hotmail 后来以 10 亿美元的天价买给了微软公司。这在当时看来有些离谱，但这样的故事到处流传，充斥着各类励志杂志，也充满着我的心。

世界上赚钱的公司多的是，但没有哪家公司的口号是改变世界。看看现在的 IT、互联网公司，每个公司的口号中几乎都包含着改变世界的梦想。而且从微软、苹果这类老一代革命家，到 Google、Facebook 这类硅谷新星，也的确有很多家公司或多或少的改变了世界，或至少是深深影响了世界。

IT，互联网，这是一个传奇的行业。当我的手抚过键盘，当眼光划过满屏自己写的代码，澎湃的激情，从来都没有息灭过。但无论多大的激情最终都会有息灭的时候。人可以依靠梦想活一段时间，但不会一直下去。1996 年，我职专毕业。职专，是差等生的代名词。职专生的梦想，一般都是成为某武侠小说的男主角，有一个酷酷的名字，有一堆惊艳的红艳。我也不例外。但这一切，在 96 年的夏天到来时，突然中止了。原因我应聘到了一个很小规模的计算机培训中心— “辰星计算培训中心”。还有，我看了辰星书柜中的电脑报合订本。

有一篇故事我仍然记得，《软件辉煌》，其中详细讲述了盖茨创办微软的经历，还有 Word 的开发历程，……。一群大男孩，用电脑改变世界。美国，硅谷，IT 界，从来都不缺这样的故事。虽然这样的故事从来没有在东方某个伟大的国家发生过。合上电脑报，一个想法出现在我的脑海中，那就是我要做中国的比尔。盖茨。年轻人总有梦想，不是吗？一年后，1997 年的夏天，四个辰星公司的“核心”员工，告别辰星，踏上创业之路。四人熟读微软、乃至硅谷的历史，这些历史让他们热心沸腾。

40 年前，晶体管之父肖克利博士的故乡，

阿里巴巴离职 DBA 35 岁总结的职业生涯 III

美国西海岸的圣克拉拉，八名怀着梦想的年轻人离开肖克利半导体实验室，创办了可谓硅谷摇篮的仙童半导体公司。这就是著名的八叛逆。而我们，则自许“四叛逆”。我们在和辰星仅一墙之隔的地方，买了二十台电脑，投资达十万，创办了红星计算培训中心。那一年，我们十八岁。

光辉岁月

——青春总是这样苦涩，但在长长生命之中，装在记忆这个坛子里，慢慢的发醇，最后却能化为无比的感纯。这或许就是生命的魔力。

……

梦想总是遥不可及

是不是应该放弃

花开花落又是一季

春天啊你在哪里

青春如同奔流的江河

一去不回来不及道别

只剩下（平凡）的我没有了当年的热血

看那漫天飘零的花朵

在最美丽的时刻凋谢

……

红星的创办，让我们的年青岁月，都达到了顶峰，此后再无超越。我们在同学中间成为传奇，成为创业的典范。甚至有同学的同学，还慕名前来，向我们请教人生。在红星除了讲课，我们还为一些传销公司开发核算、财务软件。我还用 C 和汇编，写各种有趣的程序。我用汇编写过一个 DOS 下内存驻留的程序，写完很有成就感，自信横横。但转念一想，要学的东西如此之多，不知何年何月才能学完，刹那间又充满急燥。

那段青春岁月真是难又形容。快乐与泪水，自信与自悲，坚持与彷徨，……，各种各样的感情交织着，匆匆的，过去了，不再来。寒来暑去，花儿谢了又开、开了又谢，时间就如同夏日傍晚的凉风，轻轻的带走了岁月，只给人留下，多样的回忆，哪怕这回忆是如此的沉重。小学三年级时，还是 1987 年，我在书店看到一本书，作者是诺查丹马斯。书中详细介绍了他的一系列预言，从狮心王理查的死期，到西特勒的出现。当然，最后是 1999 年人类大灾难，恐怖大王从天而降。很不巧，当时哥哥刚刚带我看完电影《超人》。超人故乡星球毁灭的特效，虽然在现在看来虽然拙劣不堪，但当时强烈震憾了我幼小的心灵。如今再看诺查丹马斯的预言，我几乎已经看到了地球在强烈的地震、大火中，变为一片废墟。出了书店，我万念具灰。我马上找到我的一众好朋友，将这个噩耗讲给他们：“我们只剩 12 年好活。”，未了，看着他们一个个脸如死灰，我却突然心情大好起来，优哉游哉的回家写作业去也。有了这种经历，使我完全相信微博上的一句话：不要向任何人诉苦，因为 20% 的人不关心，剩下的 80% 听到后很高兴。这是题外话，我们继续。

12 年后，1999 年到了，我却再无所谓。或许大家一起死了，比我迷茫的活着更好。曾经辉煌的红星，只不过几年，便已面目全非，曾经的朋友，分道扬镳。在千禧年来临的前夜……■

本文未完，请看原文：

http://developer.51cto.com/art/201204/331095.htm

淘宝下单高并发解决方案

周末参加了 @ 淘宝技术嘉年华 主办的技术沙龙，感觉收获颇丰，非常感谢淘宝人的分享。这里我把淘宝下单高并发解决方案的个人理解分享一下。

作者 / 玉开

周末参加了 @ 淘宝技术嘉年华 主办的技术沙龙， 感觉收获颇丰，非常感谢淘宝人的分享。这里我把淘宝下单高并发解决方案的个人理解分享一下。我不是淘宝技术人员，本文只是写自己的理解，所以肯定是会有一些出入的。

在 session 中牧劳为我们介绍了淘宝下单部分的技术方案变迁，我不介绍变迁，而只对现有系统做介绍。

要优化下单，提高下单的 TPS （Transaction per second），我们首先要做的是对下单的逻辑剥离，只保留核心部分，而把附加功能剔除出去。比如说下单要考虑库存量，考虑发短信，要给卖家发旺旺消息通知，要对订单做统计，要做销售额统计等等，这些功能是必要的，但是也是附加的功能，要最大程度提高下单这一步的 TPS，就要先不考虑这些东西。

下单必然会涉及到买家查看订单，和卖家查看收到的订单，修改订单价格等，这是下单的核心。 在下单这个操作中有买家和卖家两个密切关联而有不同的视角。牧劳称为两个不同的维度。据牧劳的介绍下单这一步只有 5 张表，这 5 张表涵盖了这两个维度的操作。

下单是在一个数据库事务中进行的，要提高数据库的事务并发数，最有效的办法是拆分，拆分有两种，一是对库进行拆分，另一种是在同一个库中对表进行拆分。要做拆分首先就要考虑拆分依据的字段，淘宝是根据订单号做拆分的，而下单中有两个维度，买家和卖家，对订单做拆分之后，必须还是可以通过买家，卖家方便的查询着两个维度的数据。该怎么办呢？这里留个疑问，我先介绍淘宝拆分的规模，淘宝将订单表拆分到 16 个 mysql 库中，而在每个库中又将订单表横向拆分为 64 份，相当于将一个表拆分为 1024 份。拆分之后事务会分散到 1024 套表中，这必然会很大程序上增加并发的事务处理能力（这儿我说是必然，但是淘宝在使用这种方案之前是要经过压力测试，实际测试出这种方案的 TPS 之后，才会逐步采用这种方案的）。上面留了一个疑问，经过拆分之后如何保证买家卖家快速的查询其下的订单呢？最好的办法是保证买家，卖家下的订单在一张表中，如何保证呢？淘宝的做法是将买家的 id 取模后放到订单号中。假定一个订单号是 142424594267664；这个订单号对应的订单该放在哪台服务器上的哪个表中，是根据订单的后四位 7667，对 1024 取模之后决定的；同时 7667 是买家 id 的后四位。这样买家在查询其订单时就可以通过其 id 获得其订单所在库以及表，就可以方便有效的查询买家订单了。这里会带来另外一个问题，卖家查询订单时怎么办？前面我们已经提到卖家和买家被分成两个不同的维度来做表设计，卖家

淘宝下单高并发解决方案 II

查询时不是直接查订单表，而是通过卖家维度的表来做查询。卖家维度的表的插入，更新是通过在订单插入时发一个消息来通知插入的。同样对于发短信、发旺旺也是通过消息来处理的，这些附加功能不参与到下单的事务中去。

即使这样做了库，表的拆分，依然会有问题。淘宝在双 11 时的一天的交易量就达到了 5000 多万，这样几个月过去后，这些拆分后的表中的数据量也会达到很大的一个量，处理速度就会下降。淘宝的做法是把三个月之前的老数据迁移到其他库中，这样就避免了数据量增大导致的系统响应时间降低的问题。但是会带来另外一个问题，用户在查询订单时需要同时查两个库，一个是历史数据表，另一个是近期数据表；这个问题无可避免，就是通过查询两次解决。

也许有的朋友会想到拆分之后对全数据做统计会有问题。如果在拆分后的表上做统计，是肯定会有问题的。怎么做呢？其实很简单，把数据迁移到别的库中去做统计。

表做拆分可以大大的提高 TPS，但是也会带来一些问题，需要通过可靠的消息通知机制通知其他模块做非核心处理的事情，需要通过高效的搜索系统保证搜索数据的及时更新。

以上是我个人对淘宝下单高并发设计的理解。这是肤浅的，实际做的时候肯定还需要考虑更多的问题，比如数据库的调优，磁盘 IO 方式，服务器稳定性；方案的可测试性，可量化等等。

上周六的技术还分享介绍了很多其他方面的精彩内容。感谢主办方，主持人！ 期待 @ 淘宝技术嘉年华 更多精彩的技术沙龙。

订单号介绍勘误：

文中对于订单号的表述有点问题，对于 16 台服务器，每台服务器 64 张表只需要 2 位买家或卖家 id 的后两位数字就可以准确定位到具体的库和表。订单号中同时存在买家 id 的最后两位和卖家 id 的最后两位。分别在订单号的倒数第 3，4 位数和最后两位数。

假定买家 id 为 123456789，那么在订单号中的最后两位就是 89，通过 89 对 16 取模就可以定位到具体的库上，通过对 64 取模就可以定位到具体的表上。

Java 编程开发小 TIP

在进行编程的时候最好首先建立一个空项目，然后在这个空项目里，编辑自己的 Java 程序。这样在你学习 Java 一段时间后，你就拥有了一个属于你自己的包，这对你以后学习就会很方便。

通过 Jcretor 你可以很方便的即时查阅 JDK 帮助文档！且这个功能可以实现网页直接在 Jcreator 的编辑窗口里显示，而不用另外打开任何浏览器进行浏览。(Jcreator 的这个特点我十分喜欢，据我所知 Eclipse 好象没有提供这个功能) 哈，说了这么久相信大家一定会问，你这个功能到底怎么使用了！别急，马上你也会掌握这个技巧。

当你在 Jcreator 中编写程序或看别人的原代码的时候，肯定会碰到不熟悉或不知道的类或方法，这个时候你用鼠标选中你要查询的类名或方法名，然后点击 Jcretor 菜单栏里的 Help 菜单下面的 JDK Help，这时你会发现你所要查询的资料马上就在 Jcreator 中以网页的形式显示出来了！

张利华：夺命的中国IT互联网行业

作者 / 张利华

我们每隔一段时间就会在网上祭奠一位因病去世的年轻 IT 人，真不希望这个现象继续。

刚刚过去的 4 月清明节，不少人在微博上点起蜡烛，祭奠离去的人们，这里面不乏多位 IT 界人士：既有年仅 45 岁因肺癌去世的紫光软件集团总裁，还有年仅 38 岁的智播客创始人张孝祥。今年，前华为人朋友们在网上祭奠一位刚离世不久的研发项目经理老曹，他刚领着团队完成了一个版本。他在女儿满月的晚上再也没有醒过来。“听到这个消息，确实有点悲伤。再也没人叫，宋工，老杨喊你开会了！”前华为人宋志勇辛酸地说。“同样祭奠两年前上海世博时离世的好兄弟窦宇，逝者 用生命告诫我们，身体是玩命的本钱。”多位在华为工作的网友们在微博上说。

不久前网上还疯转一最美女站长得胃癌求助床位手术的消息。是竞争过于激烈的压力？是工作过于勤奋生活无规律？是熬夜太多？是一周七天的工作令无暇锻炼？就在本文未完之时，一位曾在百度、千品网工作过的闫鹏又刚刚离世，年仅二十多岁。

珠海人林海韬 2011 年 6 月刚从中大毕业，成为“百度地图”技术工程师。11 月 14 日，他因心脏衰竭而亡。其生前发表的微博显示其工作繁忙，曾 48 小时不休不眠。

越来越多、越来越年轻的 IT 互联网界离世的生命，在提醒着人们：要把健康放在第一位！也同时在警告：IT 互联网行业即将成为一个夺命的行业，这个行业的年轻人充满梦想以及种种促进人类生活更美好的使命，强调用户的美好体验，但是为什么不先扪心自问，什么是工作的目标？工作的目的是让生活得更美好，工作以健 康为代价，何来生活得更美好。

IT 互联网界充满诱惑，吸引了各种专业的人欲改行而加入。中国 IT 互联网界有中国首富级人物，每个在国外上市的中国互联网企业都会至少诞生一位亿万富翁。 同时，IT 互联网界也是中国少数的靠自身努力而不是靠拼背景就可以步入中产阶层的行业，相对公平的创业环境吸引了大量出身平平的有志青年加入。

但是看上去发财很容易的 IT 互联网界在中国对多数从业人员并不容易。二十年前就开始引入外国先进技术的通讯设备行业在中国目前剩下华为、中兴等少数几个大企业，华为号称是中国创造最多年收入二十万以上的中产阶层的民营企业，但长期面临全球领先通讯设备企业的竞争压力，而不得不长期采取内部“压强”原则，以 求快速反应来应对激烈竞争。

相比已具备一定领先地位的华为、百度这样的大企业，处于完全竞争状态下的互联网、手机、动漫企业，每天面临的生存压力已足以令任何一位在职者不得不与时间博命。“赶版本，赶在腾讯前出版本”这是互联网界的流行话。中国手机界则面临着被苹果、三星两个大巨头挤干了市场份额和利润的境地，“一年到头白干活”。 2012 年酷

张利华：夺命的中国 IT 互联网行业 II

派、中兴等上市公司纷纷公告手机利润大幅下滑。以往每年出一款手机，现在二、三个月就要出几款新机，利润还大幅下滑，看上去种类日益繁多的手 机企业员工的压力与日俱增。中国的动漫企业多数处于亏损、生存危机中，动漫企业员工的工作时间超长，却收入超低。

面临种种压力的中国 IT 互联网企业需要反思，首先是领先的的大企业要反思：带血的第一不能要！用生命代价换取的带血的市场份额不能要！用鲜活的年轻人换来 的企业大幅增长不能要！用累死的躯体构筑的热卖新产品不能要！大企业应以负责任的态度对待将企业视为家的员工们，珍视他们的健康！不能只求发展速度，而不要员工的幸福指数。否则这样“杀鸡取卵”式取得的第一，未来也会因员工们的无法承受而再次失去，无法取得可持续发展。

创业期的 IT 企业要理性认识到，奋斗几通宵“比腾讯早出一个版本”并不能让企业不被很快干掉和生存下去，“谋定而后动”，掌握足够的策略和积累比“盲动” 更重要；而创造比大企业更灵活更宽松的工作环境、工作时间和轻松鼠氛围，让员工工作得更健康，才能充分释放创业员工的创造力，并且在薪酬无法与大企业攀比的劣势下取得稳定的人才资源。

研发效率员工搏命赶版本更重要。在苹果的影响下，2012 年华为、HTC 等多家中国手机企业表示将减少手机出品的品种，增加精品制作的投入。中国动漫产业 年产量达 22 万分钟，但只占到世界总产值的 0.68%；日本却以年产 10 万分钟的产量，占到世界动漫总产值的 68%。90 分钟的美国动漫电影《功夫熊猫 2》相当于中国 3 万分钟的动漫作品带来的产值。这些数据都表明中国动漫企业应着眼于重质量而不是赶数量。

美国同样处于竞争激烈的互联网创业企业 facebook，建议工程师们采取站着编程的方式以利于健康。

而 Google 的员工有着几乎是世界上最好的工作环境，员工们上班时间也可以随时去享用全世界最美味种类最丰富的内部餐，装饰成像家一样的个人办公空间、 宽大的休息室、完备的游乐场、豪华健身房、瑜珈室也随时欢迎员工的光顾。GOOGLE 的员工可以穿托鞋 , 短裤，踩着滑板来上班；可以在上班时间就在办公室里享受按摩的释放，或者打上几 杆高尔夫球，健健身、睡觉也很正常。员工工作区的颜色也很有娱乐的精神，有效地促进了员工的身心健康，不至于因压力巨大而产生压迫感。Google 这些以 人为本的工作方式并没有让其发展慢下来。

中国一些有紧迫感的跨国公司已经设置了首席健康官的职位。试想一下，如果一家大企业无论是曾经发生过跳楼事件，还是发生过员工疲劳死的悲剧，都能下一条死 命令：一旦再发生类似情况，分管的常务副总裁，无论业绩多好，革职勿论，这样的话，悲剧还会接二连三的发生吗？ 期待更多 IT 互网企业能设置重要人事负责的 首席健康官，切实把员工的健康与生命放在第一位。

“我每天都爬楼梯，我的办公室在 9 楼，我从不坐电梯”我一位在 IT 界工作多年的好友，虽然经常加班，但仍然保持了良好的健康。而广大 IT 从业人员也要反思：身体永远是第一位的，让生活多些滋味和健康，该放下些什么了。如果将健康放在第一位，总会有办法化解一些劳累。■

PKzip 创始人 Phil Katz 短暂而饱受折磨的一生

编者按

在新的网络精英层出不穷的年代里，我们不能忘记那些更早的技术奠基者。Phil Katz（1962–2000）是压缩软件 PKzip 的创作者……

4 月 14 日，当人们发现 Phil Katz 死去的时候，他的身体陷在 Southside 旅馆房间的一个小桌旁边，怀里抱着一个装过烈酒的空瓶子。

这位以“zip”而闻名，并且创建了一家价值数百万的软件公司的天才，最终死于长期酗酒导致的胰脏严重内出血。

他当时孤单一人，和家人已经长期断绝关系，而对于他自己创立的位于 Brown Deer 的公司 PKWare Inc. 的员工来说，他也相当于一个陌生人。

当时他 37 岁。

Phil Katz 在 1994 年的照片，手中拿着一张软盘，里边是他的公司 PKWare. Inc 发布的压缩软件。

这不是一个光彩的结局，尤其对于一个创建了世界上最有影响的软件（PKZip），不仅让技术信徒们惊叹，更让主流媒体侧目的人来说。

Katz 创作的软件，可以将计算机文件减小至 50% 到 70%，从而节约宝贵的磁盘空间。他的压缩软件为计算机行业带来了一个行业标准，并且让“zip”成为了人尽皆知的计算机术语。

但他的天才带来的财富并没有把他从后来地狱般的生活习惯中挽回来——妄想症，酗酒，脱衣舞吧。到最后，Katz 只是偶尔地进行一些工作，在很晚的时候打开电脑，而用酒精和脱衣女郎填满他的整个白天。

Katz 在 Mequon 拥有一所独立公寓，但他很少呆在那里。为躲开他的逮捕令，他在机场附近不停更换廉价旅馆居住。他通过位于 Franklin 的 Mailboxes Etc. 快递服务获得他的邮件。

“这个人在世界上连一个朋友都没有，我是说，一个真正的朋友都没有。”Chastity Fischer 说，她是经常和 Katz 在一起的一个脱衣舞女，也是最后见到过 Katz 的人之一。

“想象一下生活中一个人都没有，没有人可以打电话。没有人。”

孤独的高中

Phil Katz 是一个安静的患有哮喘的孩子，他最高的运动成就，不过是在 Glendale 附近跑跑越野摩托车。

他是 1980 年 Nicolet 高中的毕业生。早在“geek（奇客，褒义）”这个说法还没和 .com 公司联系起来的时候，他就被认为是一个 geek（怪人，贬义）了。

“他一点都不合群，而且是大家指指点点的对象。”和 Katz 一起毕业的 Rick Mayer 说，“他说话鼻音有些重。他个子矮小，而且，我不想说他不好看，所以为只能说他看上去很一般吧。”

听说 Katz 的去世之后，他高中的同桌，也是数学和物理的高材生，现在是心脏病理学医生的 Ray Fedderly 翻开他高中的毕业纪念册，在里边找到

PKzip 创始人 Phil Katz 短暂而饱受折磨的一生 II

一条带着些许不安的留言：

”我享受和你一起渡过数学和物理课的那些时光，在 Nicolet 这四个糟糕，漫长，无法忍受，折磨人而又美好的四年里。我希望你有光明的前途和幸福的生活（如果可能的话）。希望一个计算器就能为你带来莫大的快乐。”

“如果我在 18 岁时就是一个手术医生的话，我应该知道该怎么做的。” Fedderly 说，“我现在知道那是在请求帮助。而不是一个玩笑。”

作为一个天生内向的人，Katz 开始了他分析能力方面的追求。

Katz 曾经和他的父亲 Walter 在周末的下午下棋，晚上为可编程计算器编写代码，那是一个 PC 还没出现并改变计算机史的时期。

因为可编程计算器内存只有一点点，Phil 和 Walter 需要做很多非常高效的工作。

“我记得他最早写的是一个处理月球着陆的游戏程序，”Katz 以前的堂兄弟 Brian Kiehnau 说，他在 1977 年见到过 Katz。

1980 年的时候，Katz 在 Wisconsin-Milwaukee 大学主修计算机科学课程。大约在同一个时候，Walt 和 Hildegard Katz 为他买了第一台计算机，一台原始的 IBM PC，有两个软盘驱动器，一个单色显示器，还有 64k 的内存，和今日的计算机比起来微不足道的配置。

得到这台 PC 以后，Katz 立即开始在上面编写程序，将他大部分的空闲时间花在电子布告栏系统（BBS）上面，那是互联网的前身。

BBS 很快就变成了 Katz 的社交圈，在那里，别人懂得他熟练的编程技术，而且可以和他共享对计算机的激情。

渐渐地，Katz 喜欢上了在 BBS 上共享信息，因为和别人的交流可以让他把程序写得更好。可能在这些经验的影响下，Katz 后来的 PKWare 才走 上了” 共享软件 “（Shareware）这一道路。用户可以先免费尝试这个软件，如果觉得它有价值的话，再向作者付费。以 PKWare 为例，用户出 47 美 元，可以得到一份说明书和免费的升级。

“他花了很多很多时间和人们交流，帮助别人。他会在一个计算机用户组里边花几个小时和他们在一起。” Hildegard Katz 说。“他非常非常乐于助人，这是他最可爱的一点。”

但在 1981 年春的时候，悲伤席卷了这个家庭，Phil Katz 的生活就开始变得不同了。

Walter 在 55 岁的时候，由于周而复始的胸痛，进行了心脏手术。几个小时以后，他去世了。

父亲的去世 Phil Katz 是一个莫大的打击。很多年过去以后，有一次在 Katz 喝醉酒和 Fischer 聊天时还提到这个影响对他有多大。

“他父亲的去世让他心碎。有一次我和他去他父亲的墓地时，” Fischer 说，“他不停地说，如果他父亲活着的话，他们会一起去钓鱼，一起去做 ‘男人做的事情’。”

Katz 的朋友和家人说，Walter 的去世让作为儿子的他更为孤独，并且让他更加和计算机深陷在一起。

深夜编程

1984 年，Katz 毕业并且拿了计算机科学文凭，并且受雇于 Allen-Bradley Co.，他写的是运行 “可编程逻辑控制器” 的代码，这些代码，操控

PKzip 创始人 Phil Katz 短暂而饱受折磨的一生 III

着 Allen-Bradley 公司全球客户工厂车间的制造设备。

Katz 在 1986 年离开 Allen-Bradley，又开始为 Graysoft 工作，这是一家位于 Milwaukee 的软件公司。到了晚上，他会躲在卧室里写他自己的程序。

他编写的是一个 Arc 的替代程序，Arc 是当时一个常用的文件压缩工具。Katz 使用了一些算法，让程序可以将一些指定的信息合并起来，比如，将每 一个 “a-n-d” 组合从文本中提取出来，这样会消去文本中的每一个 “and”，“hand”，“sand”。这个程序可以将这些以及数以千计的其它字母 组合提取出来，并且在需要的时候将其恢复。

Katz 和他在 BBS 上的好友共享了这个软件的早期版本，称作 PKArc，而且他花了无数的时间改善这个软件。到 1987 年时，这个软件在线上已经 有了相当的影响力，以至于 PKArc 已经开始蚕食 Arc 的开发公司，新泽西的 System Enhancement Associates 的市场份额。

“我在邮件中收到一张支票，我想，哇！ 这很不错嘛，“1994 年 Katz 在 Journal Sentinel 杂志的采访中这样说，” 后来接着的几个月里，我又收到了更多的支票。“

他转向他的母亲求助。

“人们不停地打电话给他说，我们想要使用你的软件，我们要给你付钱。” Hildegard 说。

Katz 在 1987 年离开了 Graysoft，开始了他独自的奋斗。以前 Graysoft 的同事，后来 1988 年加入 PKWare 的 Steve Burg 说，PKArc 的销售已经让 Katz 在 Graysoft 的 30,000 左右的薪水显得微不足道了。

刚开始，Katz 大部分的工作都是在 Hildegard 的厨房完成的。他们雇用了电话应答服务来应对洪水般的来电，而且给了 Burg 一个程序开发员的职位。

同事们对他的才能印象很深。

“他是个及其聪明的人，” 1988 年加入 Katz 公司，直到去年 6 月才离开的 Doug Hay 说，“他记得 10 年前考试的公式，而大部分人考完后 20 分钟就忘得一干二净了。”

在后来被称为 arc 战争的那段时间里，BBS 的用户开始将他们的 .arc 压缩更换为 .zip 格式。

1988 年，System Enhancement 以版权和注册商标侵权为名起诉了 PKWare，看着诉讼费的节节升高，Katz 决定和他们达成协议。完整的协议内容不得而知，但当新泽西公司的代表遇到他们的对手时，也许吃了一惊。

“System Enhancement 的律师本来期望见到一家大公司，结果却来到了 Hildegard 的家里。” Kiehnau 说，“他在 BBS 上留了一个地址，所以律师以为那里会有一幢豪华玻璃建筑之类的东西。真的很搞笑。”

这起诉讼在 BBS 上的传播激起了人们对 System Enhancement 的大范围不满，从而加速了 arc 格式的死亡，尤其在 PKWare 很快推出新的，使用更好算法，却又不兼容 arc 的压缩工具以后。

钱像洪水般一般地涌来。■

本文未完，请看原文：

http://developer.51cto.com/art/201205/334350.htm

开发者拒绝写技术博客的常见理由

常有人跟我讨论我在 blog 上发布过的博文，有时候他们还希望我来撰写某些文章。在讨论的过程中，我几乎总是会问为什么你自己不开一个博客，或者为其他人的博客做些贡献呢？我在引导他们为技术类的主题写一些博文时，极少有人对此感兴趣。

常有人跟我讨论我在 blog 上发布过的博文，有时候他们还希望我来撰写某些文章。在讨论的过程中，我几乎总是会问为什么你自己不开一个博客，或者为其他人的博客做些贡献呢？当我在引导他们为技术类的主题写一些博文时，极少有人对此感兴趣。

我的母亲总是告诉我（以及她的学生们）每个人都有自己的故事。她说，写作可能是一种你不曾想到过的非常有益处的行为，直到你写的东西获得了其他人的喜爱和赞同。就像软件开发者为他们的用户开发软件一样，作家为他们的读者而写作。从来都没有什么论断说技术人员无法成为激励他人学习并乐于同他人分享知识的优秀作者。

我写这篇博文的目的就是想鼓励广大的技术人员去享受写作、高效的写作，并能够从中得到乐趣。我将翻出一些我曾经最常见到的关于为什么技术人员不愿意写博客的理由。



“我不认为我是某方面的专家。对于任何一个我能想到的主题，我都不是权威。”

我首先想到的是这个理由，因为它将是我最需要去驳斥的观点。如果从这篇博文里你没有得到任何收获的话，那就记住这句话：你并不需要成为某方面的专家才能去写相关的文章。

翻翻我的博客你就能找到很多这样的例子。我自认为自己是某个方面，或者某两个方面的专家，但我在过去 5 年中写了超过 450 篇博文，我肯定不会只写我最在行的那一两个主题。写下你懂得的东西，要成为权威你就不能害怕去多做些研究。这方面有一个很好的例子就是我的一篇标题为“Kerberos for haters”的博文。我几乎没有任何有关 Kerberos 方面的经验。事实上，甚至在我的 RHCA（RedHat 系统部署工程师）认证考试中我都无法正确配置好它！但是，我对此做了相当多的研究，并开始慢慢理解了这一大坨东西是如何联系起来的。还有许多人都对 Kerberos 感到困惑，于是我决定将我所掌握的有关 Kerberos 方面的知识串接起来写成一篇博文。这篇博文引来了许多正面和负面的回复，很明显我发布的博文对一些读者起到了帮助作用，启发了一些人同时也得罪了一些人。

接下来看看下一个常遇到的理由：如果我写

开发者拒绝写技术博客的常见理由 II

的东西里面有些地方是错误的怎么办？在整个互联网面前犯错，这使我看起来就像是个傻瓜。这种事我早就经历过，都已经有些厌倦了。做个不恰当的假设，每个作者都至少会犯一次错。读者们会指出你的错误（有些读者会很含蓄的指出，而有些读者不会这样），下面就是你的责任了，要么更正你的大作要么说服你的读者是他们错了。我自己也曾经发表过含有错误的博文，而且我对于时不时的要去检查这些错误显得有些懒惰。就像我中学时的新闻学老师一直教导我的：对于一个错误，最重要的部分是你要如何去更正它，并从中汲取教训。总之，你一定会犯些错误的。只要你尽心尽力的对此做研究，并尽量减少错误，及时对错误做出回应，我想读者是不会怪您的。

说到错误，这里还有一个常见的理由：我技术还不错，但我的拼写和语法很糟糕，我根本不擅长写作。这个问题很容易解决，如果你是那种什么事情都靠自己解决的人，那就拿起这本由 Strunk 和 White 合著的《The Elements of Style | 风格的要素》吧。网上有这本书的 PDF 版，或者你也可以借一本来读。无论你属于哪种情况，这本书详尽的描述了应该如何适当的加标点符号、如何组织句子和段落，以及如何正确地引用你的论点来源（对于写研究性的文章来说相当受用）。也许你并不想拿起一本如此枯燥的参考书来读，如果是这样的话，看看你身边有没有擅长写作的人。在大公司里你常常能在市场部或者公司的通讯录里找到一些愿意读你的文章并帮你做批注的伙计（谢谢你 Garrett！，感谢你曾经为我修订过文章）。在 Fiverr（著名的 5 美元服务区，这是一个在线的信息平台，特色之处在于其收集那些价值 5 美元的事情，用户可以发布、接受或者分享这些信息）上我甚至找到了一些愿意以 5 美元的价格为我修订文章的人。

下面我们将看看这第二常见的理由：我不知道该写给谁看。如果我写的东西太简单了，那些真正的技术大牛们会不会觉得我就是个小白？如果我写的东西超级复杂，大多数人都接受不了又怎么办？

这些我都经历过。大多数称职的 Linux 系统管理员都知道该如何添加和删除 iptable 规则，他们会觉得这是相当简单的任务。但是你知道吗，在我写过的超过 450 篇的博文里，这篇《deleting a single iptable rule》的访问量每个月都是排名前 5！我每个月的点击量里有 11% 都来自这篇博文。人们要么通过这篇博文学到了一些知识，要么是他们忘记了该如何删除一条 iptable 规则，想通过这篇博文做快速的参考。不管如何，这篇博文对于许多人来说都是很有价值的，尽管在我看来这个主题相当的简单。反过来说，我曾经发疯写过一个完整的关于云主机冗余配置的 how-to 类文档，结合了 LVS、glusterfs、MySQL on DRBD、memcached、haproxy 以及 Idirectord 等各种技术。我觉得对于某些读者来说这个文档会很有价值，但肯定会搞晕绝大部分读者。事实证明我又错了，这篇博文长期占据我博客的前 10 名受欢迎的文章之一，通过这篇博文我收到的回复、email 和 IRC 比其他的文章都要多。■

本文未完，请看原文：

http://developer.51cto.com/art/201205/333433.htm

编者按

有时候，能了解表面背后的故事，知道一些程序编译过程，其实对写出一个正确的正确的程序会有很大的帮助，这起源自一段很简单的问题实现：两整数交换。

画虎画皮难画骨，编程编码难编译

古语“画虎画皮难画骨”，是说画老虎时要画它的外表很容易，可要将老虎的气势画出来却很难。

对于现在的程序员来说，似乎也是这样子，可以写出整洁的代码，设计出优异的程序，但却不一定需要知道代码在编译之后的是如何运行的。

但是有时候，能了解表面背后的故事，知道一些程序编译过程，其实对写出一个正确的正确的程序会有很大的帮助，这起源自一段很简单的问题实现：两整数交换。

简单的两数交换不简单

最早学习到的实现方式，就是使用临时变量来保存一个数进行交换：

```
int main()
{
int a = 21;
int b = 7;
int tmp = a;
a = b;
 b = tmp;
return 0; }
```

后来不经意间又学习到了不用临时变量，只用算数运算符加减就可以进行交换的方式：

```
int main()
{
int a = 21;
int b = 7;
a = a+b;
b = a–b;
a = a–b;
return 0;
}
```

这个问题在思想上讲十分的巧妙，不过在计算机的世界里这并不是一个好的方法，考虑一点编译后的问题，我们就会碰到溢出的问题，导致得到错误的结果。

高手们又使用位运算的异或运算来消除临时变量的方法，还不用担心溢出的问题，于是我又学习了：

```
int main() {
int a = 21;
int b = 7;
a ^= b;
b ^= a;
a ^= b;
return 0;
}
```

画虎画皮难画骨，编程编码难编译 II

甚至有更简短地把三个异或运算写成一行的：

```
int main() {
int a = 21;
int b = 7;
a ^= b ^=a ^= b;
return 0;
}
```

就像我在《这些没有可读性的代码，却又体现出程序员对语言的高度理解力》一文里想表达的，这种写法体现了书写者对运算符顺序的深刻理解，对异或运算符特殊性的充分了解。

两数交换方法的编译后的指令

虎皮画得很精美，根根毛发都细致描绘，但是没有看过老虎骨骼和肌肉，可能就感受不到万兽之王从骨子里散发出的那种无畏的霸气。

没有看过编译后的代码，我才会总不能理解为何那么简短优美还“节省空间”的代码怎么就没有成为一种标准写法呢？好像只是作为偶然拿出来炫给初学者们，让他们惊讶和眼睛一亮的把戏。

在网上搜索了一下有关这方面的问题，在stackoverflow 上看到了这样的一段回复：

> Using this XOR might actually be slower than the usual two value swap using a temp variable; the fact that you are writing through references suggests you are forcing the compiler to do 3 memory (ok, cache line) writes, whereas temp-style swamp should only do two memory write (any intelligent compiler will put the temp in the registers). Adding the conditional check surely makes it slower. So while this might be fun, it probably isn't a practical thing to do in a sort.

从这句话看出，似乎看出算数法和异或法虽然节省了一个微不足道的空间，但是却会浪费时间。效率上还不及临时变量法，更不要提其中出现的如溢出等隐蔽的问题了。

在学习了一点 gcc 指令的入门之后，我使用了

```
Ider$ gcc -S swap.c
```

来观察了一下从 C 文件编译出来的汇编指令的代码，才发现：虽然在 C 种交换都只用了三行，但是编译后的汇编指令行数却不尽相同。

对比三种方式的 main 的汇编指令：

（每一个注释表示指令所对应的 C 代码，一行 C 代码可能对应多个汇编指令，从注释行开始计算，到下一个注释行之前那行都属于那行 C 代码对应的指令）

Temp

```
_main:
Leh_func_begin1:
pushq   %rbp
Ltmp0:      movq    %rsp, %rbp
Ltmp1:    movl    $21, -12(%rbp)    ;;int a = 21
movl    $7, -16(%rbp)       ;;int b = 7
movl    -12(%rbp), %eax     ;;int tmp = a
movl    %eax, -20(%rbp)
movl    -16(%rbp), %eax     ;;a = b
movl    %eax, -12(%rbp)
movl    -20(%rbp), %eax     ;;b = tmp
```

画虎画皮难画骨，编程编码难编译 III

```
movl    %eax, −16(%rbp)
movl    $0, −8(%rbp)        ;;return 0
movl    −8(%rbp), %eax
movl    %eax, −4(%rbp)
movl    −4(%rbp), %eax
popq    %rbp
ret
Leh_func_end1:
```

Arithmetic

```
_main:
Leh_func_begin1:
pushq   %rbp
Ltmp0:
movq    %rsp, %rbp
Ltmp1:
movl   $21, −12(%rbp)    ;;int a = 21
movl    $7, −16(%rbp)      ;;int b = 7
movl    −12(%rbp), %eax    ;;a = a+b
movl    −16(%rbp), %ecx     addl   %ecx, %eax
movl    %eax, −12(%rbp)
movl    −12(%rbp), %eax    ;;b = a−b
movl    −16(%rbp), %ecx     subl   %ecx, %eax
movl    %eax, −16(%rbp)
movl    −12(%rbp), %eax    ;;a = a−b
movl    −16(%rbp), %ecx     subl   %ecx, %eax
movl    %eax, −12(%rbp)
movl    $0, −8(%rbp)        ;;return 0
movl    −8(%rbp), %eax
movl    %eax, −4(%rbp)
movl    −4(%rbp), %eax
popq    %rbp
ret
Leh_func_end1:
```

XOR

```
_main:
Leh_func_begin1:
pushq   %rbp
Ltmp0:
movq    %rsp, %rbp
Ltmp1:
movl    $21, −12(%rbp)     ;;int a = 21
movl    $7, −16(%rbp)      ;;int b = 7
movl    −12(%rbp), %eax    ;;a ^= b
movl    −16(%rbp), %ecx     xorl   %ecx, %eax
movl    %eax, −12(%rbp)
movl    −16(%rbp), %eax    ;;b ^=a
movl    −12(%rbp), %ecx     xorl   %ecx, %eax
movl    %eax, −16(%rbp)
movl    −12(%rbp), %eax    ;;a ^= b
movl    −16(%rbp), %ecx     xorl   %ecx, %eax
movl    %eax, −12(%rbp)
movl    $0, −8(%rbp)        ;;return 0
movl    −8(%rbp), %eax
movl    %eax, −4(%rbp)
movl    −4(%rbp), %eax
popq    %rbp
ret
Leh_func_end1:
```

（对于一行的异或运算交换法，得到汇编指令跟三行的是一样的。）

从汇编指令的行数，可以看出正如stackoverflow上讲的一样，利用临时变量的代码要明显短于其它两个。

对于临时变量法，每次赋值只要读取一个变量的值到寄存器，然后再从寄存器写回到另一个变量中即可。■

本文未完，请看原文：

http://developer.51cto.com/art/201205/334076.htm

在.NET平台上使用Scala语言(上):初尝

Scala 是 Java 平台上的一门新兴起的语言,我也不止一次在博客上提到它。我非常希望它可以取代 Java 这种劣质语言,让 Java 平台的生产力上一个台阶。



Scala 是 Java 平台上的一门新兴起的语言,我也不止一次在博客上提到它。我非常希望它可以取代 Java 这种劣质语言,让 Java 平台的生产力上一个台阶。事实上,Scala 从一开始——或者说“很早”就对生成 .NET 程序集提供了支持。只可惜,毕竟还是 Java 平台的东西,.NET 方面的资源少之又少,不过我们还是可以进行一番尝试的。

有人经常把 F# 和 Scala 进行比较——可能都是新语言吧,毕竟 C# 和 Java 之争也有很长的历史沉淀了。不过,我倒不认为两者有什么可比性。F# 是一门“拥有面向对象特性的函数式编程语言”,而 Scala 是一门“拥有函数式编程特性的面向对象编程语言”。事实上,从设计思路来说,Scala 与 C# 或 Java 是同根的,与 C# 在函数式编程方便更有可比性——虽然它在追求语言灵活性的道路上走得有些极端。因此,我认为 Scala 可以定位作 Java 语言的替代品,而 F# 却不是 C# 的替代品。因为一旦 F# 替代了 C#,便几乎代表函数式编程范式替代了面向对象编程范式。

Scala 语言的名字取做“Scalable”,暗指这是一门“伸缩性”强的语言。为什么说它伸缩性强呢?我的理解是它希望可以做几乎“任何形式 / 风格”的程序设计。这点并非玩笑,Scala 的确在走这条路。Scala 的语言非常灵活,语法的“变化”非常复杂。我一直说,这门语言实在是太甜了——到处是糖,而且可谓“无所不用其极”。Scala 似乎希望在一切可以节省的地方,如括号,标识符,进行节省。还例如,Scala 的 import 支持引入整个包,包的一部分,整个包但排除某个类,甚至在方法内部引入成员。再例如成员的访问级别,除了典型的 public, protected, private 之外,它还引入了“当前对象”这种访问级别(private 是相同“类”)。

但是,要知道 Scala 最终还是编译为 Java 平台上的二进制格式,也就是说 Scala 的功能事实上是受平台限制的。因此,使用 Scala 一些高级功能编译出的结果,其实是无法被 Java 语言体现出来的——这也是我为什么会说 Scala 的定位像是“取代” Java,而不是与 Java 共存。

至于我尝试 Scala.NET 的原因在于:

- 熟悉 Scala 语言。目前有一些有趣的,有用的类库是用 Scala 实现的,相信以后也会越来越多。
- 借 Scala 语言及 .NET 平台与 Java 平台的差异。
- 观察 Scala 一些语言特性在 IL 上的表现——毕竟 .NET 领域对于 IL 的追求■

本文未完,请看原文:

http://developer.51cto.com/art/201204/331942.htm

在.NET平台上使用Scala语言(下):分析

上一篇文章里我们简单尝试了在 Scala 里编写 .NET 应用程序。这个过程并不困难,因为似乎 Scala 官方已经对此已经有较好的支持了。我们要做的只是“获取工具”,“编译成 IL”,最后……



上一篇文章里我们简单尝试了在 Scala 里编写 .NET 应用程序。这个过程并不困难,因为似乎 Scala 官方已经对此已经有较好的支持了。我们要做的只是“获取工具”,“编译成 IL”,最后再“生成程序集”即可。那么,这些工具究竟做了些什么,Scala 究竟又是如何支持 .NET 平台的,它的可用性究竟如何,我们还需要进一步的分析及尝试。

现在看第一个问题。我们知道从 Scala 源代码生成 IL 文件的脚本是 scalac-net.bat。如果需要了解它做的事情,最直接的方法莫过于查看其中的内容。如果要看明白它的代码,可能需要我们对 cmd 命令有些了解——不过我也只是略知一二罢了,如果您对其了解不多其实也没有太大关系。经过合理推测,我们知道 scalac-net.bat 本身不会有什么功能,它只是调用编译器而已。因此,这个脚本文件的职责,无非是收集参数并执行编译器。于是我们打开 scalac-net.bat,在众多 for/if 之中可以发现它最后执行了这样一个命令:

```
%_JAVACMD% -Xbootclasspath/a:"%_BOOT_CLASSPATH%" %_JAVA_OPTS% %_PROPS% -cp "%_EXTENSION_CLASSPATH%" scala.tools.nsc.Main -target:msil %_ARGS%
```

那么我们再调用 scalac-net.bat 的时候这行命令究竟是什么呢?对于此类问题,我们可以再它前面加上 ECHO 命令,即:

```
ECHO %_JAVACMD% -Xbootclasspath/a:"%_BOOT_CLASSPATH%" %_JAVA_OPTS% %_PROPS% -cp "%_EXTENSION_CLASSPATH%" scala.tools.nsc.Main -target:msil %_ARGS%
```

ECHO 可以视为 cmd 的 print 命令,我们可以用它来观察和学习脚本。再次运行,便可以看到编译器的调用方式了:

```
D:\scala-2.7.7.final\code> ..\bin\scalac-net.bat test.scala  java -Xbootclasspath/a:"D:\SCALA-~1.FIN\bin\..\lib\scala-library.jar" -Xmx256M -Xms16M -Dscala.home="D:\SCALA-~1.FIN\bin\.." -Denv.classpath="" -Dmsil.libpath="D:\SCALA-~1.FIN\bin\..\lib\predef.dll;D:\SCALA-~1.FIN\bin\..\lib\scalaruntime.dll;D:\SCALA-~1.FIN\bin\..\lib\mscorlib.dll" -Dmsil.ilasm="c:\Windows\Microsoft.NET\Framework\v2.0.50727\ilasm.exe"  -cp "D:\SCALA-~1.FIN\bin\..\lib\mscorlib.dll;D:\SCALA-~1.FIN\bin\..\lib\predef.dll;D:\SCALA-~1.FIN\bin\..\lib\sbaz-tests.jar;D:\SCALA-~1.FIN\bin\..\lib\sbaz.jar;D:\SCALA-~1.FIN\bin\..\lib\scala-compiler.jar;D:\SCALA-~1.FIN\bin\..\lib\scala-dbc.jar;D:\SCALA-~1.FIN\bin\..\lib\scala-library.jar;D:\SCALA-~1.FIN\bin\..\lib\scala-swing.jar;D:\SCALA-~1.FIN\bin\..\lib\scalaruntime.dll" scala.tools.nsc.Main -target:msil test.scala
```

可以看出,这是在运行一个 java 程序,并且提供了很多参数。不过参数很多,内容也很乱。不过乱的原因在于其中对于各式命令或者库文件的引用都使用的完整路径。■

本文未完,请看原文:

http://developer.51cto.com/art/201204/331947.htm